100% AMIGA

PROGRAMMING

Amazing COMPUTING™ For The Commodore AMIGA®

Your Original AMIGA® Monthly Resource

Volume 4 No. 10 October 1989
US $3.95 Canada $4.95

PUTTING IT ALL TOGETHER

Programming:

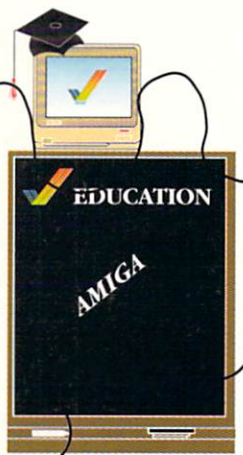
APL And The *Amiga*
Saving 16-Color Pictures In BASIC And True BASIC
A Designer Language Explored: Multi-Forth
Using Gadgets In Assembly
Function Evaluator In C

Reviews:

SimCity
JForth Professional
Hi-Soft Compiler

Hardware Project:

Better TrackMouse



Education:

Big Machine On Campus:
The Amiga At The CSU Summer Arts Program

The Amiga Attends Rutgers

Reading, 'Riting, and Resolution:
Graphics Programs With Kids In Mind

Typing Tutor:
A BASIC Typing Tutorial Program

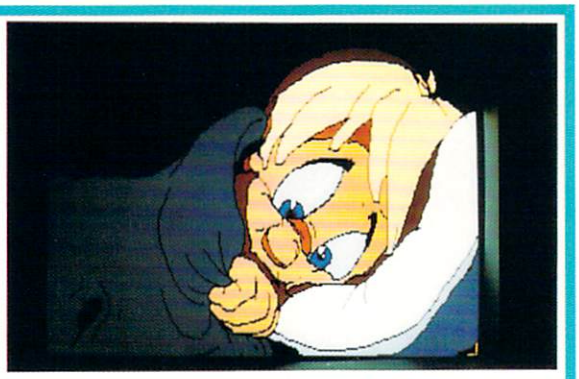
10
0 74470 74710 9

Amazing COMPUTING™ For The Commodore AMIGA®

Volume 4, Number 10
October, 1989

Amazing Report:
Education and the Amiga

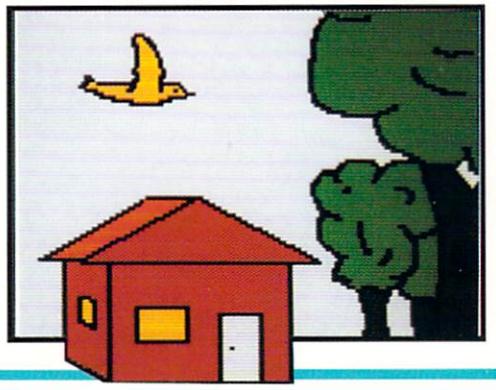
Big Machine on Campus 8
by Joel Hagen
Humboldt State University in Northern California goes Amiga!



Commodore's Educator! 17
CBM uses video to reach the classroom.

Typing Tutor 18
by Mike "Chip" Morrison
Save the city of Keycaps from capital letters.

Reading, 'riting & resolution 14
by Joe DiCara
Three paint programs primarily designed for pre-school and elementary grades.



Roomers 53
by the Bandito
The Bandito lists his annoyances, while Commodore gets ready for Christmas.

Video Schmideo 56
by Barry Solomon
AC's Video Editor gives us helpful hints and tips in video!

C Notes from the C Group 82
by Stephen Kemp
A look at a search utility program.

COLUMNS

New Products and Other Neat Stuff 24
by Elizabeth G. Fedorzyn
Populous, OMNI-PLAY Basketball, and more!

Snapshot 31
by R. Brad Andrews
Help Rambo save Colonel Trautment in Rambo III, or drive a Ferrari in the Grand Prix.

PD Serendipity 35
by Mike Morrison
Mike reviews Fred Fish disks #229 to #236.

No Fishing! 38
by Graham Kinsey
Graham reviews PD programs from the local BBS.

Bug Bytes 50
by John Steiner
John keeps us up-to-date with the latest software bugs!

REVIEWS

HiSoft Compiler 29
by Cole Calistra
A BASIC compiler that has good features, is quick, and well worth the price!

SimCity Review 41
by Miguel Mulet
How would you handle Boston, Massachusetts right before a major nuclear meltdown?

•TABLE OF CONTENTS•

JForth Professional 85
by Jack Woehr
It fills the requirements of the professional Forth programmer.

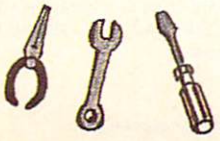
AMAZING KNOW-HOW

Better TrackMouse 22
by Robert Katz
A true one-handed trackball mouse!

CONFERENCES

SimCity Conference 43
edited by Richard Rae
A conference with Will Wright and Brian Conrad of SimCity fame.

A1000 Rejuvenator 47
edited by Richard Rae
A conference with Gregory Tibbs.



PROGRAMMING

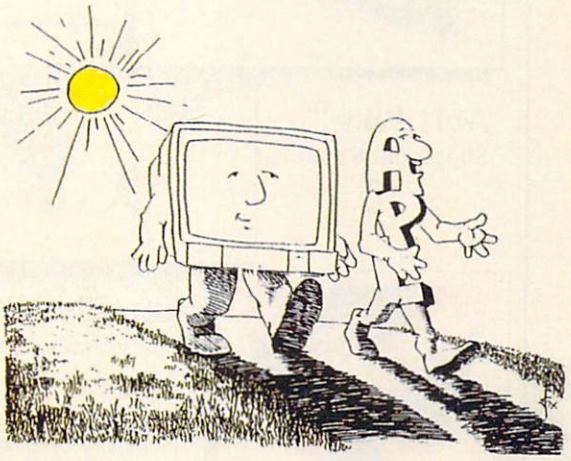
APL & the Amiga A Friendly Pair 19
by Henry Lippert
A look at the hidden language of APL.

Saving 16-color pictures in high-resolution 58
by Paul Castonguay
Part Three of the Fractals series.

Multi-Forth 68
by Lonnie Watson
How to implement an interface to the ARP library.

More Requesters in AmigaBASIC 71
by John Wiederhirn
Pushing beyond the limits of BASIC with system routines.

Glatt's Gadgets 88
by Jeff Glatt
Adding gadgets in Assembly.



Tshell Part II 93
by Rich Falconburg
A new Amiga program that will enhance the command line environment.

Function Evaluator in C 99
by Randy Finch
A routine that accepts mathematical functions as string input and then evaluates the function.

SIDEBAR

The Amiga In Higher Education 11
by Tony Preston
Rutgers University, in New Jersey, recognizes the special value of the Amiga.

DEPARTMENTS

From the Managing Editor 4

Amazing Mail 6

Index of Advertisers 96

Public Domain Software Catalog 109

Amazing COMPUTING™

For The Commodore AMIGA™

ADMINISTRATION

Publisher:	Joyce Hicks
Assistant Publisher:	Robert J. Hicks
Circulation Manager:	Doris Gamble
Asst. Circulation:	Traci Desmarais
Asst. Circulation:	Donna Viveiros
Corporate Trainer:	Virginia Terry Hicks
Traffic Manager:	Robert Gamble
International Coordinator:	Marie A. Raymond
Marketing Manager:	Ernest P. Viveiros Sr.
Administrative Support:	Aurore R. Trepanier

EDITORIAL

Managing Editor:	Don Hicks
Associate Editor:	Elizabeth Fedorzyn
Hardware Editor:	Ernest P. Viveiros Sr.
Technical Editor:	J. Michael Morrison
Music & Sound Editor:	Richard Rae
Video Editor:	Barry S. Solomon
Copy Editor:	Aimée B. Abren
Copy Editor:	Derek J. Perry
Copy Editor:	Karen Donnelly-Solomon
Art Director:	William Fries
Photographer:	Paul Michael
Illustrator:	Brian Fox
Production Manager:	Donna M. Garant

ADVERTISING SALES

Advertising:	Jannine Irizarry
Special Assignment:	Barry Solomon
Marketing Assistant:	Melissa J. Bernier

1-508-678-4200
FAX 1-508-675-6002

SPECIAL THANKS TO:
Buddy Terrell & Byrd Press
Bob at Riverside Art, Ltd.
Swansea One Hour Photo
Pride Offset, Warwick, RI
Mach 1 Photo

Amazing Computing™ (**ISSN 0886-9480**) is published monthly by PiM Publications, Inc., Currant Road, P.O. Box 869, Fall River, MA 02722-0869.

Subscriptions in the U.S., 12 issues for **$24.00**; in Canada & Mexico surface, **$36.00**; foreign surface for **$44.00**.

Second-Class Postage paid at Fall River, MA 02722 and additional mailing offices.

POSTMASTER: Send address changes to PiM Publications Inc., P.O. Box 869, Fall River, MA 02722-0869. Printed in the U.S.A.

First Class or Air Mail rates available upon request. PiM Publications, Inc. maintains the right to refuse any advertising.

Pim Publications Inc. is not obligated to return unsolicited materials. All requested returns must be received with a Self Addressed Stamped Mailer.

Send article submissions in both manuscript and disk format with your name, address, telephone, and Social Security Number on each to the Associate Editor. Requests for Author's Guides should be directed to the address listed above.

Amazing Dealers

The following are **Amazing Dealers**, dedicated to supporting the Commodore-Amiga™. They carry *Amazing Computing*™, your resource for information on the Amiga™. If you are not an Amazing Dealer, but would like to become one, call.

1-508-678-4200

Alaska
Juneau Electronics — Juneau
That Computer Place — Anchorage
Alabama
Abax Data Systems — Huntsville
Alabama Book Store — Tuscaloosa
Madison Book & Computer — Madison
Mel's Photo & Computer Shop — Montgomery
Pasco Industries — Birmingham
The Computer Image — Birmingham
The Computer Nut — Birmingham
Universal Computer System — Mobile
Arkansas
Commodore Connection — Sherwood
Microtronix — Fort Smith
The Micro Shop — Little Rock
Arizona
Bookman's — Tucson
Books Brothers Ltd. — Tucson
Books Etc. — Tempe
Brain Child Software — Phoenix
Computer Library, Inc. — Tempe
Livingston's Books, Inc. — Sierra Vista
North American Digital — Tucson
Try Tech Inc. — Pheonix
California
3 E Software And Systems — Hayward
Aday International — Foster City
American Mags & Books Ltd. — Englewood
Arcadia Newsstand — Arcadia
AugNET — Santa Clara
B. Dalton Books — San Rafael
Balboa Office Supply — San Diego
Bits & Bytes — Fairfield
Bits n Pieces Comp Prod. — Anaheim
Bookland — Marysville
Bookstar — San Diego
Brentano's — Sacramento
Brentano's — San Francisco
Brown Knows Computing — Redlands
CAL-Tech Bookstore — Pasadena
Candy Computer — Elk Grove
Castro Kiosk — San Francisco
Centerfold News — Los Angeles
Central Park Bookstore — San Mateo
Century Computer Systems — La Habra
Circus of Books — Los Angeles
Circus of Books — Los Angeles
Commodore Land — Alhambra
Commotech — National City
Computer Attic — Palo Alto
Computer Literacy Books — Sunnyvale
Computer Literacy — San Jose
Computer Literacy — Santa Clara
Computer Nook, Inc. — San Bernadino
Computer Outlet — Chula Vista
Computer Outlet — San Marco
Computer Room — Scotts Valley
Computer Showcase — San Francisco
Computer Spectrum — Burlingame
Computer Time — Citrus Hts
Computers Plus — Grove City
Connet Newsstand — Oceanside
Covina Bookstore — Covina
Creative Computer Systems — El Cajone
Dataphile — Watsonville
Del Amo Books & News — Torrance
Delaurel News Agency — Oakland
Discount Computer Station Inc. — Long Beach
ECX Computer Co. — Walnut Creek
Empire Computers — Santa Rosa
Encinitas News — Encinitas
FI-American — San Francisco
Fisher Computers + Software — Rancho Cucamonga
Harding Way News — Stockton
Heinrichson, Schneider & Young, Inc. — Berkeley
Home Computing Center — Concord
Home Computing Center — Mill Valley
Home Computing Center — San Leandro
HT Electronics — Sunnyvale
Int's Periodical Dist — San Diego
International Telecine — Santa Ana
K J Computers — Granada Hills
L. A. Int's News — Venice
Little Professor Books — Placentia
LTC Computers — Santa Ana
Lyon Enterprises — Berkeley
Lyon Enterprises — Berkeley
Micro Right — San Jacinto
Micro P. Technologies — Lawndale
NewsSeat — Davis
Norma McCay — San Diego
Northridge Newsstand — Northridge
Op Amp Tech Books — Los Angeles
Paras Newsstand — San Diego
Parsin International — San Diego
Printers Inc. — Palo Alto
Put's Electronics — North Highlands
R&R Software — Chula Vista
Ridgecrest Computer Ctr. Inc. — Ridgecrest
S.O.S. Computers — Los Angeles
Sawyer's News Inc. — Santa Rosa
Software Cellar — Anahiem
Software Central — Pasadena
Software First — Santa Rosa
Software First — Vallejo
Software Plus — Citrus Heights
Software Solutions Plus — El Cerito
Stacy's #5 — Modesto
Technitronics Service — South Pasadena
The Bookmith — San Francisco
Third Avenue News — Chula Vista
Tower Books — Chico
Tower Books — Citrus Hts.
Tower Books — Concord
Tower Books — Mountain View
Tower Books — Sacramento
Tower Books — San Mateo
Universal News — Hollywood
Vintage Wine — San Diego
Warner Engineering — San Diego
WCI Computer Central — Eureka
Winners Circle — Berkeley
World Book & News — Hollywood
Colorado
Aurora Newsstand — Aurora
Citadel Computers — Colorado Springs
Computer Discount — Denver
Computer Repeats — Boulder
Computer Room — Aurora
Micro Computer World — Fort Collins
Micro World — Lakewood
Paragon Computers — Colorado Springs
Paragon Software & Electronics — Colorado Springs
Sun Country Computers — Colorado Springs
Tattered Cover Bkstore — Denver
Westerminster Newsland — Westminister
Connecticut
Aetna Life Club Store — Hartford
Amigo Business Computer — Norwalk
Beta Press — Derby
Computer & Software Connections — Plantsville
Demc Electronics — Hamden
Friendly Computers — Plantsville
G & G Christ Church — Derby
Gordon & Gotch — Derby
Inkotronics, Inc. — Woodbury
Int'l Mag Dist/Capetown — Derby
Lac Distributors — Norwalk
Lebanon/Levant Dist. — Derby
Liberty Book — Derby
Mnemonics Inc. — Norwalk
One Byte — Quaker Hill
Personal Computer Center — Norwich
Software City — Orange
Software Kingdom — East Hartford
Tricom Computers — North Windham
District of Columbia
Programa — Washington
World News — Washington
Delaware
Castle Video — Newark
Newark Newsstand — Newark
Florida
A A Computers — Jacksonville
Ami Comp Computer Center — Oviedo
Bookstop — Clearwater
Bookstop — Miami
Bookstop — Orlando
Brentano's — N. Palm Beach
Chuck's Newsstand — Vero Beach
City Newsstand — Melbourne
Commodore Country Inc. — Pinellas Park
Computer Bar — Pennsacola
Computer Base — Brandon
Computer Image — Miami
Computer Lane Inc — Pembroke Pines
Computer Room — Ft. Lauderdale
Computer Specialties — Melbourne
Computers + Video Creations — Palm Beach
Computers 4 Rent — W. Palm Beach
Computers Etc. — Sarasota
Computers Plus — S. Daytona
Creative Equipment — Miami
Education Computers Etc. — Talahasse
Extra Extra — Tampa
Florida Book Store — Gainesville
Gen-Rod — Miami
Gulf Coast Computer Exchange — PanamaCity
Joe's News, Inc. — Miami
Malvar Freight — Miama
Mark Twain Smoke Shop — Lakeland
Micro Computer Center — Largo
New Age Electronics — Clearwater
New Age Electronics — St. Petersburg
PC Collection — Coral Gabels
Program Store — Clearwater
Program Store — Pinellas Park
Prospect News & Books — Ft. Lauderdale
Public Domain solutions — Sarasota
Software City — Orange Park
Sunny's At Sunrise, Inc. — Sunrise
Sunshine Software — Ft. Lauderdale
The Open Door — Merritt Island
Georgia
AmpexSystem — Norcross
BFJ Enterprises — Roswel
Brentano's — Atlanta
Cardinal Computers — Dalton
Micro Center/Marietta — Marietta
Mostly Software — Columbus
Oxford Books, In.c. — Atlanta
Personal Computer Exchange — Lilburn
Software City Savannah — Savannah
Software South Inc. — Savannah
The 64 Store — Atlanta
Hawaii
Astro Computers & Electronics — Aiea
Computronics — Honolulu
Data-One Computer Center — Honolulu
J & C Repair — Aiea
Idaho
A.B.I. Computer & Video — Nampa
Illinois
Audio Visual Req. — Chicago
Books & Bytes, Inc. — Naperville
Computer Software Connection — Moline
Comp—Kingdom — Lisle
Coopersmith — Vernon Hills
Digital World — Naperville
DigitalWorld — Addison
DOS Computer Center — Niles
FD Software & Cadds — Bloomingdale
Keep In Pace Computers — Champaign
Kroch's & Brentano's — Auroa
Kroch's & Brentano's — Chicago
Kroch's & Brentano's — Orland Park
Kroch's & Brentano's — Schaumburg
M Two J Two, Inc. — Champaign
M2J2 Inc. — Champaign
Memory Expansion — Bloomington
Micro Age — Oaklawn
Micro Ed Enterprises Inc. — Chicago
Micro Tech Soultions — Aurora
Micro Tek — Kankakee
Ring Software — Geneva
Software City — Bellville
Software Plus Chicago — Chicago
Software Plus West — Hanover Park
Software Plus — Wheeling
The Memory Expansion — Bloomington
Unique Computer — Sterling
Warren Radio/FRANCO — Peoria
Indiana
Bookland — Indianoplis
Burkat Computer Ctr. — South Bend
Cal Com Inc. — Kokomo
Computer Corner — Ft. Wayne
Computer Corner — Marion
Computer People Inc. — Michigan City
CPU Inc. — Indianapolis
Indiana News — Indianapolis
Micro Computer Centers Inc. — Indianapolis
Programs to Go — Muncie
Reese Equipment Co. — Plymouth
The Book Corner — Bloomington
The Computer Room of Merrrville — Merrville
Vons Computers — West Lafayette
Xerox Computer Center — Indianapolis
Iowa
Computer Solutions & Facts — Des Moines
Computers & More — Des Moines
Gronert Computers — Des Moine
Soft Search — Ames
Kansas
Arc Electronic Distributors — Wichita
Brentanos — Mission
Micro Media — Hays
Midwest Computers — Manhattan
Software City Overland — Overland Park
Software City — Overland Park
Software Exhange of KS — Wichita
Kentucky
Computer Dimensions Inc. — Lexington
Computer Mania Inc. — Florence
Expert Services — Florence
Hawley Cooke — Louisvile
Hawley Cooke — Louisvile
Reality World Computers — Lexington
Softech Computer Systems — Lexington
The Computer Lab — Louisville
Louisiana
Bookstar — New Orleans
City Newsstand — Baton Rouge
Computer Time — Shreveport
Lakeside News — Metairie
Modern Business Machines — Metairie
Software Center International — Metairie
Software Source — Lafayette
The Book Shelf — Slidell
Maine
Computer Barn — Damariscotta
Hands On Computing — Westbrook
Skylight Software — Belfast
Maryland
Buried Treasure — Rockville
Cal Com Inc. — Silver Spring
Capital Classics Ltd — Silver Spring
Greetings and Readings — Towson
Knowledgeware Systems — Kensington
Micro Computer Center — Baltimore
New Age Computers — College Park
Professional Micro Services — Baltimore
Software Advantage — Rockville
Software City — Rockville
Waldorf Computer — Waldorf
Massachusetts
Computer Works — Middletown
Computer Zone — North Attleboro
Dcx — Williamstown
HCS Computer Center — Marshfield
LCA Video & Computer Ctr. — Norwood
Memory Location — Wellsley
Newsbreak, Inc. — Swansea
OmniTek Computers — Tewksbury
Pioneer Valley Data Equipment — Amherst
Soft Designe — Watertown
Softpro — Burlington
Software Connections — Swansea
The Bit Bucket — New Newton
The Software Haus — Quincy
Tycom Inc. — Pittsfield
Michigan
Borders Book Shop — Novi
Central Park Computers — Okemos
CompuSoft — Kalamazoo
Computer Emporium — Lansing
Coopersmith's — Livonia
Direct Access — Lansing
Global Computer Center — Dearborn
I/O Softwware — Flint
Little Professor Books — Brighton
Michigan News Agency — Kalamazoo
Michigan Software — Novi
Micro World — Livonia
Micro World — Milford
Micro World — Roseville
Micro World — Southfield
Micro World — Southgate
Micro World — Sterling Heights
Brands Mart Computers — Kansas City
Carthage Electronics — Carthage
Clark Worldwide — Kansas City
Data Byte Computer Co. — St. Louis
Software City — Independence
Systems Plus of St. Louis — St. Louis
Systems Plus — Overland Park
Nebraska
Double E Electronics — Omaha
My Com Inc. — Plattsmouth
Nevada
Computer House — Sparks
Computer World — Las Vegas
Nat'l Discount Computer — Las Vegas
Tower Records — Las Vegas
New Hampshire
Dartmouth Bookstore — Hanover
Diversified Computers — Keene
System Eyes — Merrimack
New Jersey
Act One — Vineland
Atlantic News — Atlantic City
Business System Professional — Scotch Plains
Enachpea Enterprises — Fort Lee
Family Computer Centres — South Orange
Garden State News — Cliffside Park
Hudson News — New York
Morris County Stationers — Chester
Program Store — Princeton
Software Station — Rockaway
The Program Store — Eaton Town
The Program Store — Wayne
Unlimited Software Inc. — Rutherford
Village Computer Center — Cedar Knolls
Wilson Group USA — Jersey City
New Mexico
New Horizon Computer System — Alamogordo
Newsland Bookstore — Albuquerque
Page One Newsstand — Albuquerque
Road Runner — Espanola
New York
Amagination — New York
Amcorp Ltd — Huntington Station
Amicus- Hudson Valley — Wappingers Falls
Amiga Business Computer — East North Port
Book Distributor — Huntington Station
Broadway News — Saketzga Springs
Byte Shop — Merrick
Compute Center — White Plains
Computer Metropolis — New Paltz
Computer Outlet — Jamestown
Computers Etc. — Syracuse
Computerware — East Meadow
Contemporary Computing — Latham
Cromland Inc. — Brooklyn
Global Software — New Hyde Park
Island Software Computer Services — Coram
Kemco Publisher Ser — East Northport
Leeder's Video — Schenectady
Leigh's Computers — New York
MCP Associates — Astoria
MicroWorks — Kenmore
Cincinnati Fin SQ News — Cincinnati
Computer Network — Newark
Eddy's Cigar Store — Elyra
Fairborn Home Computer — Dayton
Infinity Computing — Columbus
Lakes Consumer Electronics — Akron
Lakewood International News — Lakewood
Leo's Book & Wine Shop — Toledo
Local Computer Store — Chesterland
Medical Computers — Centerville
Micro Center-Columbus — Columbus
Micro Center-Westerville — Westerville
Microtech Computer Center Inc. — Toledo
Microware Magic — Fairfield
News Shop — Kent
North Coast Programming — Willoughby
Not Just Software — South Toledo
Not Just Software — West Toledo
Paper & Ink — Tiffin
PC Powerhouse, The — Fairfield
Power Computing Inc. — Toledo
Salem Computer Center — Salem
Software Centre International — North Olmsted
Software and More — Cincinnati
Software Connection — Mayfield Hts.
Somers Photo & Comp. — Cleveland Heights
Upgrade — North Olmstead
Oklahoma
Little Professor Books — Stillwater
Mert Computers — Oaklahoma City
Micro Link Computer Store — Tulsa
Second Hand Software — Oklahoma City
Video-Comp Inc. — Lawton
Oregon
Andromeda — LaGrande
Clackamas Computers — Beaverton
Clackamas Computers — Clackamas
Comm-Shack — Salem
Libra Books Inc. — Eugene
News & Smoke Shop MF — Medford
PC Express — Eugene
Powell's Tech — Portland
Rich Cigar Store — Portland
Sixth & Washington News — Portland
Soft Cell Software, Inc. — Portland
Software Pipeline-Salem — Salem
Software Pipline — Beaverton
Tech Books — Portland
Pennsylvania
Alpha Omega EBE — Warren
ATD Software — Sayre
Basic Computer Systems — Erie
Basic Computer Systems — Hermitage
Bryn Mawk News — Bryn Mawk
Circle Computer — Ephrata
Clark Worldwide — Middletown
Computer Dimensions — Atoona
Computer Wurx — Stroudsburg
Computers and Games — Reading
Downington Computer Center — Downington
Elmo's Computers — Brownsville
Floppy House Software — Palmyra
Gene's Books — King of Prussia
Graphics Computer Center — Lionville
Lauer-Todd Computer Solutions — Newtown
Micro Master Inc. — Monaca
North Central Digital Systems — Danville
Pittsburgh Computer Store — Pittsburgh
Software Hut Inc. — Philadelphia
Some Hole In The Wall — Philadelphia
T A Slick & Co, Inc — Lancaster
Tasco Computer Center — Lancaster
The Floppy Disk — Mechanicsburg
Triangle Computers — Indiana
Walden Hill Images — Berlin
Rhode Island
Micro Limits — Smithfield
Software Connection — Warwick
The Computer Place — Woonsocket
South Carolina
Avcom International Inc. — West Columbia
Software Haus — Charleston Hts
Watson's Jewelry — Chesterfield
Tennessee
Bristol Micro Technolog — Bristol
ComputerLab of Memphis — Memphis
Computer Hardware Services — Nashville
The 64 Store — Bartlett
Texas
A&E Computers — Grove
All Systems Go Software — San Antonio
Bookstop — Arlington
Bookstop — Austin
Bookstop — Dallas
Bookstop — Houston
Bookstop — San Antonio
Brentano's — Dallas
Byte Shop — Austin
C.E.M. Corporation — Brazoria
Computer Ease — Corpus Christi
Computer Magic — Austin
Computer Skills — Eules
FNS Computer Services — Texarkana
Maxwell Book Inc. — Desota
Metropolitan Computer — Richardson
Micro Search — Houston
Software Library — Witchita Falls
Software Terminal — Fort Worth
The Computer Experience — San Antonio
The Computer Stopl — Amarillo
The Federated Group — Mosquite
Trinity News Company — Fort Worth
Utah
Armadillo Bros — Salt Lake City
Magazines and Gifts — Salt Lake City
People's Computers — Provo
Vermont
Ray Supply Inc. — South Burlington
Virginia
Audio Fidelity Corp — Richmond
Brentanos — Alexandria
Discover — Fairfax
Family Computer Center — Fairfax
Family Video & Computer Center — Newport News
I/O Computer — Norfolk
Tract Software — Prince William
Video Computer Retailers — Chesapeak
Virginia Micro Systems — Woodbridge
Washington
A.P.P.L.E. CO-OP — Renton
Amazing Electronics — Spokane
Apple Valley Computers — E. Wenatchee
Bits, Bytes and Nibbles — Spokane
Brentano's — Seattle
Bull Dog News — Seattle
Byte World — Kent
Computer Quest — Edmonds
Computers + Inc. — Tacoma
Family Computers — Everett
Micro Age Computer Store — Bellevue
Nybbles & Bytes — Tacoma
Omni International — Seattle
Port Book and News — Port Angeles
Steve's Broadway News — Seattle
Swan Computers — Yakima
Take A Byte — Kennewick
Tech Star — Kent
The Computer Shoppe — Vancouver
Tower Books — Bellevue
Tower Books — Seattle
Tower Books — Tacoma
Y Not Magazines and Books — Federal Way
Wisconsin
20th Century Books — Madison
Computer Software Center — Milwaukee
Fox Valley Personal Computer — Oshkosh
Madison Programmer's Guide — Madison
Mayfair Computers — Milwaukee
Mom's Computers — Stevens Point
Software City — Racine
TMW Software Inc. — Wausau
Wyoming
Byte Emporium — Laramie
Computer Works — Laramie
Crazy Mountain Elect — Casper

Outside of the United States

Canada
Alberta
Coles, Book People — Calgary
Coles, Book People — Edmonton
Software Supermart — Edmonton,
British Columbia
Computime Computer & Software — Prince George
Stricter Computer Centres — Delta
Unihall — Vancouver
Manitoba
Adventure Software, Ltd. — Winnepeg
New Brunswick
Computer World — Monoton
P.G. Compusystems Inc. — Campbellton
New Foundland
J.W. Allan Co. Ltd. — St. John's
Nova Scotia
Atlantic News — Halifax
Coles, Book People — Yarmouth
Kobetek Systems — Halifax
Renmatt Enter Ltd. — Greenwood
Ontario
All Computers — Gloucester
Arbon Electronics Ltd. — Toronto
Brittons Smoke Shop — Otawa
Canadian Software Source — Downsview
Chatham Computer Sales — Chatham
Classic Computer — London
Coffey's News Depot — St. Thomas
Coles Bookstore — Rexdale
Coles, Book People — Quelph
Coles, Book People — Scarborough
Coles, Book People — Toronto
Coles, Book People — Willowdale
Comput-or Software — Barrie
Computer & You, The — Islington
Computer Express — Mississauga
Computer Force — Sudbury
Computer House — Chatham
Computer Odyssey, Inc. — Hamilton
ComputerLand — Cornwall
Computroniks — Brampton
Computronix Data Centre — Toronto
Comspec — Toronto
Con-pute — Oshawa
Cryston Computer System — Midland
CVF Data Systems — Peterborough
Daily Planet, The — Whitby
Data Marketing Kingston — Kingston
Desktop Computer, Inc. — Kitchener
Electronics 2001 — Willowdale
Foreign Correspondence — Kingston
Globe Magazine — Otawa
Gran Bookstore, The — Toronto
Granger T.V. Ltd. — Godench
Hi-Land News — Barrie
Horizon Computers — Whitby
Humber College Stores — Rexdale
Kahle Electronics — Kanata
Lichman's — Thornhill
Lichtman's — Willowdale
Lightman's News & Books — Toronto
Litchtman's — Toronto
Lynx Technical Services — Otawa
Mags & Fags — Ottawa
Mikadon Computer Engineering — Thunder Bay
Mr. Diskettes — Otawa
Novatech — Niagra Falls
Nu-Info Teck — Ottawa
Peach Micro — Otawa
Personal Computers — London
Personal Keyboard Systems — London
Prospero Books Ltd. — Otawa
Smoker's Corner — Pickering
Smoker's Place — Colborne
Smokers Plus — St. Catherines
The Software Project — Hamilton
U of W Bookstore — Waterloo
World's Biggest 200 — Toronto
Zoez Software — Brantford
Zoez Software — Milton
Quebec
Azmut Infor. Inc. — Quebec
Camelot-Info — Montreal
E. M. Press & Books, Inc. — Montreal,
Jupiter Technologies Inc. — Ft-Claire
Logicel International — Montreal
Multimicro — Ville D'Anjou
Ordinateurs 2000 Inc. — Hull
Quebits Logicel Inc. — Montreal
Star Tek Infosolutions — St. Leonard
Saskatchewan
CD's Computer Connection — Saskatoon,

Distributed in Canada by
Ingram Software Distributors — Concord
Compulit Distributors — British Columbia

Argentina
Logic Computer — Buenos Aires
Australia
Computermart — Leederville,WA
Diskworks — Greenwich, NSW
John Verhoeven — Cloverdale, WA
Maxwell Office Equipment — Abbotsford, Melburn
Run Soft — Melbourne
Saudia Arabia
Abdulla Fouad & Sons — Daman
Scotland
The Amiga Center of Scotland — Edinborough
roemwich, NSW
John Verhoeven — Cloverdale, WA
Maxwell Office Equipment — Abbotsford, Melburn
Run Soft — Melbourne
Saudia Arabia
Abdulla Fouad & Sons — Daman
Scotland
The Amiga Center of Scotland — Edinborough
Sweden
Nicholas Peerson — Lund
Procomp

Microworld Computer Center — Dearborn
Midland Computer Shop — Midland
Midwest Micro Computer — Lapeer
New Horizons Bookshop — Roseville
Norman Business Control Sys. — Wyoming
Readmore Newscenter — Lansing
Rochester Book Center — Rochester Hills
Roseville Computer Store — Warren
Slip Disk — Madison Heights
Software City — Grand Rapids
Software House — Kalamazoo
Specialists Computer — Livonia
State Street Computers — Ann Arbor
T.J. Enterprises — Clinton
The Edge Connector — Clio
Ye Olde Computer Shoppe — Ypsilanti
Minnesota
Brentano's — Minneapolis
Computer Excercise World — Minneapolis
Computer Stuff — Minneapolis
Computers, etc. — Eagan
JMH Software — Maple Grove
Look Ahead Software — Northfield
Shinder's — Burnsville
Shinder's — Crystal
Shinder's — Edina
Shinder's — Inver Grove Hgts
Shinder's — Maplewood
Shinder's — Minneapolis
Shinder's — Minnetonka
Shinder's Read More — Minneapolis
Shinder's — Roseville
Shinders — St. Paul
Specialists IN — Baline
Valiant Inc. — Hopkins
Wizard's Computer Shop — Stillwater
Mississippi
National Computer Center — Crystal Lake
Missouri
Assoc Computer Services — Ocean Springs

Mnemonics — Springfield
QB Software, Inc. — Brentanos
Readers Market — North Port
Rochester Books — Forest Hills
Sherry Tunkel — Tonawanda
Singapore Book — Rochester
Software & Such — New York
Software City — Huntington Station
Software House — Scotia
Software Link — Spring Valley
Software Supermarket II — Fairport
Star Tech Systems — White Plains
Synfaire Trading — Cheektowaga
Sysgut Computers — Massena
The Computer Cellar — New York
The MicroWorks — Albany
Unicorn Electronics — Tonawanda
Video Computer Bin — Johnson City
Video Computer Center — Brooklyn
Village Computer Inc. — Rome
World Computers Inc. — Hicksville
North Carolina
Digitz — Raleigh
K&S Newsstand — Clemmons
K&S Newsstand — Kernersville
K&S Newsstand — Winston-Salem
News Center — Cary
Parker News Inc. — High Point
Servatronic Technologie — Burlington
Software City — Charlotte
Software City Fayettevl — Fayetteville
Software City — Raleigh
Triad Computers — Winston-Salem
North Dakota
Computer Associates — Fargo
G. F. Computers — Grand Forks
Software House — Fargo
The Computer Store — Minot
Ohio
B5 Software — Columbus
Brentanos — Cincinnati

Become An
Amazing Dealer
TODAY!

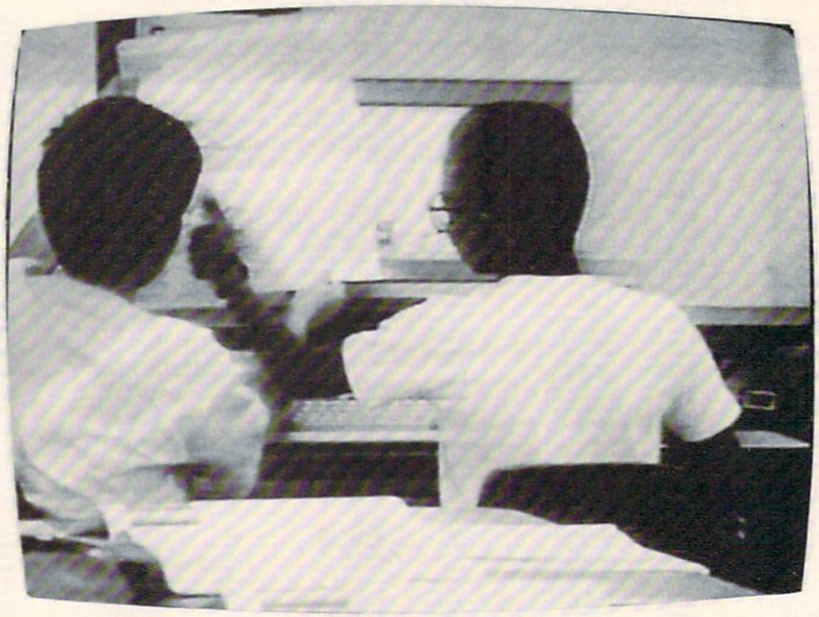
Amazing Computing™ is also available in most B. Dalton Booksellers, B. Dalton Software Stores, Crown Books, Software Etc., selected WaldenBooks Stores, and Walden's Software Store locations.

From The Managing Editor

Commodore Takes the Amiga to Education

Computers and education have long been a strong combination. Once the personal computer was available to the general public, educators quickly began using them in instruction as well as administrative roles. The first educational computer programs were little more than drill routines and examinations. The students responded to a series of random questions and the results were graded. If the students passed the lesson's required skill level, they were presented with the next lesson and if they failed to achieve an adequate score, they were returned to the lesson for review and re-testing.

Later, other aspects of the computer's ability were used. Graphics and sound were introduced to reward correct answers. Soon programs were created which avoided answers completely and provided instruction through stories and interaction. Now, Commodore is focusing the Amiga at even greater educational needs.



Commodore's Amiga Video, Amiga—Creativity in the Classroom
The Amiga is presented as an extension of a student's creativity.

Commodore and Education

At one time, Commodore commanded a large portion of the educational computer market. The Commodore Pet was a pioneer in the quest for "computer literacy." However, due to time, economy, competition, and internal company policies, Commodore lost this edge. In the last year, Commodore Business Machines has launched a multipart program of advisors, grants, special promotions, and video tapes aimed directly at the educator. Now, it is apparent that CBM wants to regain their position, and they believe the Amiga is the way to do it.

Commodore began by creating an Educational Advisory Board consisting of professors, teachers, principals, educational dealers, and business executives. Commodore went further by creating their Desktop Video Grant Program which awarded an Amiga 2500 with a hard drive, genlock, monitor, and a variety of software to each of its twenty winners.

This spring, Commodore began a special promotion directed at educators. They bundled an Amiga 500 with memory expansion and a 1084 monitor for $999.00. They also announced that Amiga 2000 systems and Amiga 2500 bundles would be offered at a special educator's discount.

Now Commodore has released a video tape aimed to excite the educational market in the potential and cost effective use of the Amiga in education. (See the article "Commodore's Educator" on page 17). In the video, Commodore demonstrates the Amiga's use in several grade levels and aspects of education.

One of the most amazing aspects of this video is what it lacks. All of the operations and advancements demonstrated by the students are performed with commercial software packages. None of the students are shown programming the Amiga. And, although the Amiga is heralded throughout the video as an advanced and powerful computer, little is said about the actual hardware configurations. The entire line of Amiga computers, from the Amiga 2500 to the Amiga 500, is shown without demonstrating their differences.

The effort by Commodore is obviously to showcase the Amiga's abilities in a classroom environment without scaring the potential Amiga educator with a technical presentation. The video is constructed to show the Amiga as a tool for creative expression instead of a computation device. While the Amiga is certainly an advanced tool for art, music, video, and education in general, it is unfortunate that the video did not stress the computing languages and materials available to students to manipulate and create their own programs on the Amiga.

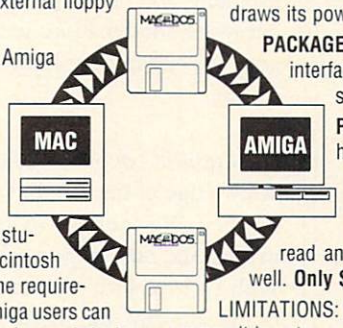
Most people feel the strength of a computer in an educational environment is based on how many educational extensive programs exist for that machine that are educational extensive. In this video, Commodore has shown that the best use of a computer in an

MAC 2 DOS™

Mac-2-Dos lets you read and write Macintosh diskettes on your Amiga!

Mac-2-Dos gives your Amiga the power to read and write files to and from 400k and 800k Macintosh floppy disks using a standard Macintosh-compatible 3.5-inch external floppy disk drive connected to your Amiga.

Here are a few typical Mac-2-Dos uses: ✔ Amiga users can now have access to the extensive variety of Macintosh clip art available on Macintosh disks! ✔ Amiga users can now take their Amiga PostScript files (on a Macintosh diskette) to most any typesetting service bureau to be output on professional typesetting equipment! ✔ College students who are required to have a pricey Macintosh can now choose the Amiga and still meet the requirement of being Macintosh compatible! ✔ Amiga users can transfer all kinds of files, like word processing and desktop publishing files, spreadsheet files, or database files. ✔ Musicians can quickly and easily transfer Standard Midi Files (SMF) between the Macintosh and Amiga!



Mac-2-Dos includes a custom hardware interface, driver software, file conversion software, and, optionally, a Mac-compatible 3.5-inch floppy drive. The hardware interface plugs into the Amiga external disk drive connector or into the last external drive of the daisy-chained disk drives. The Mac drive draws its power from the Amiga.

PACKAGE A: Package A includes a custom hardware interface, file transfer software, and file conversion software. **Only $99.95***

PACKAGE B: Package B includes a custom hardware interface, file transfer software, file conversion software, a Mac-compatible 3.5-inch floppy drive, and a software driver to allow the Mac drive to be used to read and write standard AmigaDOS diskettes as well. **Only $349.95†**

LIMITATIONS: Mac-2-Dos is a disk file transfer utility program; it is not a communications program, nor is it a Macintosh emulator. It DOES NOT permit Mac programs to run on the Amiga.

* Plus $3.00 shipping/handling, † Plus $5.00 shipping/handling
CO residents add appropriate sales tax.

QUARTERBACK™

The FASTEST Hard Disk Backup Utility!

"Many in the Amiga community consider Quarterback V2.2 the standard of our community.... My overall impression of Quarterback is smooth and complete.... For the average Amiga harddrive owner, Quarterback is still the utility of choice."

- Steve Dock, ***The Amiga Sentry****, March, 1989*

"...Quarterback is the program I've chosen to keep my hard disk backed up.... Given the added power and lower price of Quarterback, it would be my first choice for a hard disk backup program."

- Matthew Leeds, ***Commodore Magazine****, June, 1989*

"The fastest of the lot, Quarterback V2.0 copied my 4.3 megs to five floppies in four minutes and fifty seconds. The user interface is smooth and intuitive, the process painless and reliable, and the speed a good sight better than even second runner up.... There are just enough gadgets around to make things easy, and not so many as to make them confusing. This is the program we use to back up our hard disk and Bernoulli data." *- Mark R. Brown,* ***INFO #26****, May/June, 1989*

Only $69.95 Plus $3.00 shipping and handling, CO residents add sales tax.

Coming Soon! **QUARTERBACK TOOLS**
A collection of high-quality user-friendly utilities.

DOS 2 DOS™

Transfers MS-DOS and Atari ST files to and from AmigaDOS!

It transfers both binary (pure data) and ASCII (text) files.
Let DOS-2-DOS be your PASSPORT to the world of foreign disk formats.

Only $55.00
Plus $3.00 shipping and handling, CO residents add appropriate sales tax.

CCS™

Central Coast Software
424 Vista Avenue
Golden, Colorado 8040
Phone 303 / 526-1030
FAX 303 / 526-0520
Dealer Inquiries Welcome

MasterCard VISA

Circle 145 on Reader Service card.

educational background is to use the computer as a tool.

By allowing students to perform applications in the pursuit of other tasks, the students have not only learned how to use the computer and continued their education, but they have grown in the concept that the computer is a tool. The computer becomes an appliance of learning rather than the subject of lessons.

Education and You

Why should education be of interest to all Amiga users? The reasons Amigans purchased an Amiga are as diverse as the individuals themselves. Each of us look at the computer and see different applications. For most individuals, education alone is not a prime inducement.

We do learn with the Amiga. Like the students in Commodore's presentation, we grow each day by applying the Amiga. By working on individual problems and utilizing software and hardware combinations to achieve a goal, we expand our knowledge and ability.

Each project becomes easier with the knowledge of the last, but we are not satisfied. We continue to improve by making the next project more complex. With each project, we use more features of the software and hardware available. In time, we gain new abilities. We grow and, as the early Amiga advertisements said, the Amiga does provide the creative edge.

Going the next step

It is comforting to know we are becoming more familiar with the potential of the Amiga. However, there are ways to be even better Amiga users. You can learn more about your Amiga and advance Amiga use in total by doing one simple thing, help a kid.

Children are the hope and future for all of us. They posses one aspect of knowledge that we have tried so hard to lose, their innocence. There is not a single Amiga user who would not benefit from working with his brother, daughter, cousin, or neighbor on the Amiga.

Besides the obvious opportunity of playing the latest arcade game under the guise of "demonstrating" the Amiga, children can ask questions and provide a perspective on things we are sometimes too busy to notice. Children see things differently and want to understand totally. Once excited, a child is completely involved in a project.

Young people are problem solvers and a computer will encourage them more than anything else. The one-to-one relationship between a young person and the Amiga allows a young Amigan to see instantly whether they are on the right track. Nothing is more exciting than watching a child overcome a difficult problem. Their success becomes our success. Soon they begin teaching us.

An Amiga Education

Education on the Amiga is more than a long list of "educational programs". It is the ability to learn and develop through using more and more tools. It doesn't matter whether you are using a word processor, a spreadsheet, a paint program, or an adventure game (a great tool for teaching logic and mapping skills), each time you or your friends (young and old) sit down to the mouse and keyboard, you are on an adventure. You start with a task and end with an achievement. And, through your efforts, have enriched yourselves.

•AC•

From "Amiga—Creativity in the Classroom" *Students attempt a variety of tasks on the Amiga.*

SupraDrive™

Simply The Best Amiga Hard Drive.

Before you buy a hard drive, look around. Look closely. Compare speeds, but also look at Interfaces...Software...Value. We think you'll agree that SupraDrives are Simply The Best Amiga Hard Drives. Here's why ...

SUPRADRIVE HARD CARD FOR AMIGA 2000
with revolutionary new
WORDSYNC™ INTERFACE

Each SupraDrive for the A500/A2000: Autoboots directly from FFS partition • Interface allows super smooth video, sound, etc., with no rude interruptions for hard drive access • Compatible with Bridgeboard™, RAM, digitizers, other boards • Supports MS-DOS disk partitions with Bridgeboards • Installs easily • Pre-formatted & ready to use • Includes 2 floppy disks of software & clear, thorough manual • Fine tuned assembly language driver software • Blind data transfers improve speed 3x • Knowledgeable tech support • Interfaces also available separately

SUPRADRIVE FOR AMIGA 500
with **AMIGA BUS PASS-THROUGH**
for unlimited expansion •
OPTIONAL 2MB RAM

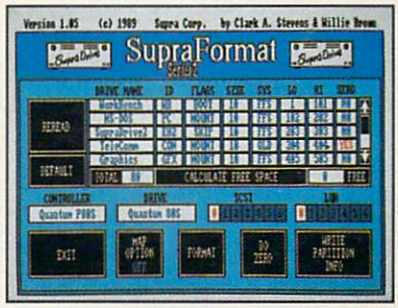
All Supra Products Are Made in the U.S.A.

Breakthrough Speed.

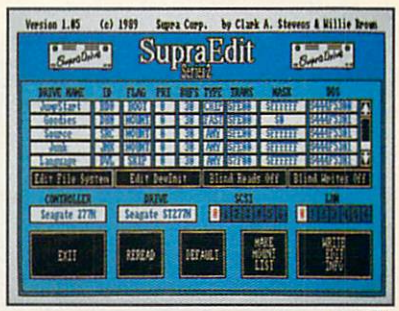
SupraDrives give you access times as low as 11 ms. and data transfer speeds of over 500K/sec. (Amiga® 2000) or 326K/sec. (Amiga 500). Features like full support of Workbench™ 1.3, the Fast File System (FFS), and multitasking make these drives *FLY*.

State-of-the-Art Interfaces.

Supra's interfaces (included with every SupraDrive) give you innovative features no one else can match. The revolutionary WordSync™ Interface transfers 16 bits at once, which gives A2000 SupraDrives DMA speed without DMA hassle. The A500 interface passes the Amiga bus signal through to your other peripherals; without Amiga bus pass-through, your system is severely limited. And all Supra interfaces feature SCSI ports for easy daisy-chaining.



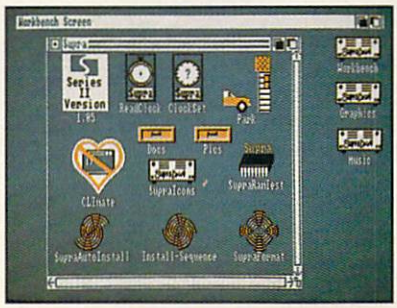
SupraFormat™



SupraEdit™

The Best Software.

After installing the drive, you'll be glad you have Supra's full array of powerful, easy-to-use software. SupraFormat makes formatting a breeze and lets you use up to 30 partitions and various file systems — FFS, MS-DOS®, Unix, Macintosh™, and more! SupraEdit lets you access low-level Amiga system information, and other included programs make using a hard disk fun and easy.



Included Software

Irresistible Value.

All this is available at a price you'll love. Look into it! Only Supra Corporation — an experienced company with a proven commitment to the Amiga and its potential — gives you such an attractive alternative: The SupraDrive. It's Simply The Best Amiga Hard Drive.

Ask your dealer for details, or call:

Supra Corporation

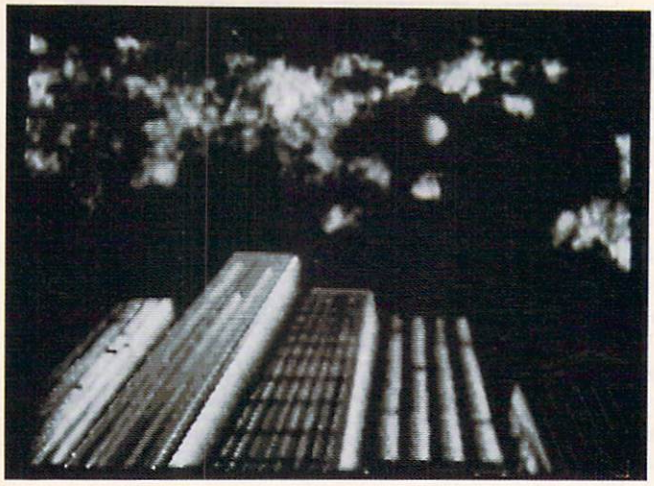
1133 Commercial Way Albany, OR 97321 503-967-9075

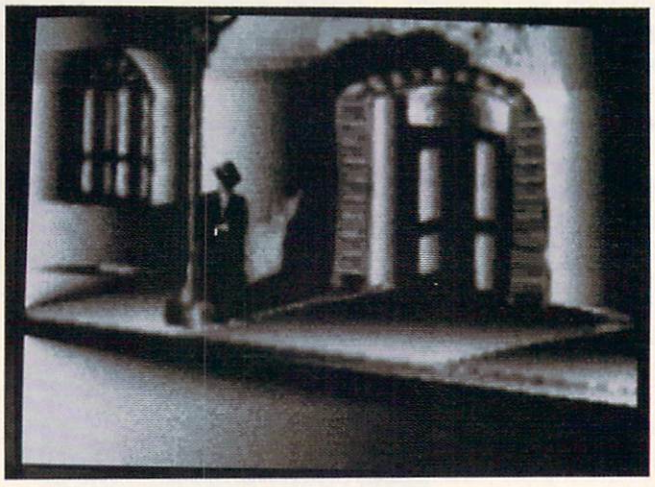
ORDERS: 1-800-727-8772

Big Machine On Campus

The Amiga plays a major role at the CSU Summer Arts program

by Joel Hagen





Like a Robert DeNiro movie: ominous scenes from, "The Hit", an animation by Neal Stillman.

Imagine an Amiga class taught by 3D and animation maestro Steve Segal, digitizing and rotoscope expert Gene Brawn, and performance professionals Rob Terry and Jody Gillerman. Twenty-five students participated in that class this past July, turning out remarkable work in marathon sessions. I was the fifth Amiga specialist teaching there, specializing in paint techniques.

The class was part of the largest interdisciplinary arts program in the Western United States. Humboldt State University in the Northern California coastal redwoods was home to the fourth year of the CSU Summer Arts program. Each summer, the nineteen campuses of the California State University system pool their efforts and their budgets to allow students to "perfect their artistic skills through interaction with world-renowned professional artists," as the catalogue puts it. For four weeks, the campus is alive with activity from electronic music to chamber groups, dance to opera, video to figure drawing. Outside, audiences were also privileged to see many of these guest artists on stage in the spacious auditorium. During the four weeks, hardly a day went by without a public performance or lecture.

Initiation

The computer graphics class is always extremely popular. Students have a wealth of equipment at their disposal, including sixteen Amigas, six color Mac II's, and four or five Targa systems running Tips and other graphics software. There were laser printers, HP Paintjets, scanners, cameras, digitizers, framegrabbers and video editing systems. A full interactive performance room was set up with Amigas, a Fairlight board, LIVE!, a MindLight 7 system, and Mandala. The students never had fewer than three instructors, and at times as many as seven. Monitors were glowing virtually around the clock, and it was not uncommon for instruction to still be going on after midnight.

The man responsible for initiating this program is Chico State art professor Steve Wilson. In addition to his work in traditional media, Wilson has a penchant for electronics and was building his own computers before most artists were even aware of them. With the advent of sophisticated, affordable computers and graphics software, Wilson is a persuasive advocate for their infusion into the arts. He sees the computer as a means of accelerating learning, as a superb tool for tinkering with ideas, for enjoying the immediacy of experimentation. His suggestion to us for conducting classes was to bombard the students with tools, techniques and ideas; give them more than they could absorb with the idea of making them aware of the myriad of possibilities that exist for an artist using

A3001

BREAKING THE SPEED BARRIER

New 25Mhz 68030 Upgrade Kit from GVP Puts Tomorrow's Performance in Your A2000 Today

SPEED 25Mhz 68030 CPU and 68882 FPU with fully implemented BURST mode 32-bit memory. An order of magnitude faster than A2500.

POWER Up to 8MB of State-of-the-art NIBBLE MODE 32-bit DRAM memory averages zero waitstates during 68030 BURST mode.

EFFICIENCY Built-in Hard Disk Controller direct on 32-bit bus keeps ALL A2000 expansion slots free even when our hard disk is installed!

QUALITY Backed up by GVP's unique full one year warranty.

A3001, 25Mhz 68030/68882 Upgrade Kit Technical Highlights:

- Asynchronous 68030 design allows high clock rates and GENLOCK compatibility.
- Factory installed 25Mhz 68882 floating point processor, twice as fast as the older 68881 chip.
- Fully DMA-able AUTOCON-FIGured 32-bit wide memory. Up to 8MB.
- Built-in AUTOBOOTing hard disk controller supports up to two 40 or 80MB drives (11/19ms access).
- Selectable 68000 fall-back mode for full floppy-based game compatibility.

GVP

GREAT VALLEY PRODUCTS INC.

Circle 158 on Reader Service card.

For more information, or for your nearest GVP dealer, call today. Dealer inquiries welcome.

FAX (215) 889-9416 • (215) 889-9411 • BBS (215) 889-4994

225 Plank Ave., Paoli, PA 19301

computers. We surrounded them with tools and software, showed them how to use them, and stayed with them into the night to answer questions one-on-one as they took off with their projects. The results were impressive.

The secret of the event's success

The success of this program is probably due to many factors. In the first place, the course is well designed by Steve Wilson. The students are highly motivated, and have chosen to attend this program rather than spend their summer another way. This is a far cry from the average junior college student looking for three easy units to slide through. The caliber of talent was high. Because so many more students apply for admission to the class than can be accepted, an initial filter is achieved by examining slide portfolios. Many of the students are working professionals in the arts. Of great importance was sufficient hardware. Each student had a computer available day and night, no waiting. Computer projects seem to require long blocks of uninterrupted time. This need was filled at Humboldt with the flat-out intensity of a "studio" atmosphere, instead of the sluggish time-clock creativity of the classroom.

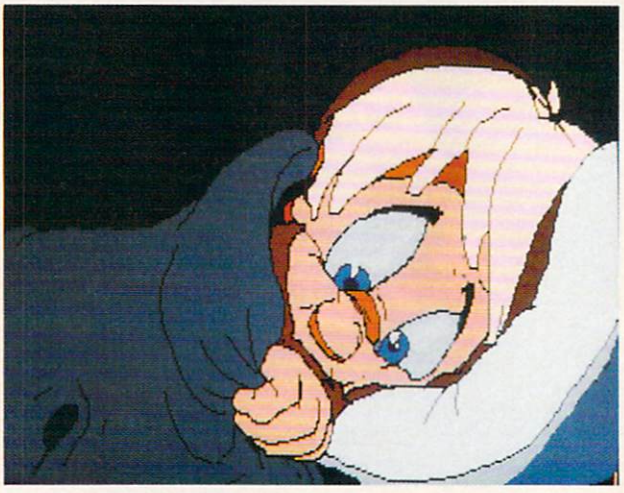
As a long time freelance artist, I want to emphasize this last point. A studio atmosphere is elusive. At times, it happened at Humboldt. When creative people are working together in one room, trying new techniques and exchanging ideas, an interesting environment can emerge. Good ideas propagate. Fatigue, with camaraderie, becomes intensity. Stress becomes a sundance and marvelous solutions emerge from mundane problems. More than once with the students, I saw the Guru trash a day's work only to see a superior version emerge by sunup. I often saw one student's good idea picked up by others and played like visual jazz, variations on a theme. Also, students helped each other, teaching techniques learned earlier in the day. I think there is no better way to imbed new knowledge than to immediately teach it to someone else. As it happens, I am talking about a summer arts program. I could as easily be talking about a user group or just a handful of local Amiga enthusiasts who decide to get together and jam every couple of months. There is always plenty of solo time. Occasional full immersion in that elusive studio atmosphere pays rich rewards.

Perhaps the lab situation on campuses could be altered a bit to foster this atmosphere. Many computer labs are mixed graphics and business. Loud brainstorming around the Amigas by art students disturbs the business student typing his term paper. A special graphics night might be established in these labs with extended hours for late nighters. Enough computers to accommodate several students is, of course, critical to the flow of learning. This brings up one of the Amiga's obvious advantages—its low cost. For instructional purposes, the combination of price and power is unbeatable. For a student headed toward a career as a professional graphics artist, solid experience on the Mac is essential. It is the standard tool in most graphics firms. But for instructional purposes in everything from design to video, painting, animation and ray tracing, I don't think there is a better choice than the Amiga. One student, an art teacher by profession, was particularly excited about teaching color theory on the Amiga. He saw it as a way of getting right to the concepts without having students bogged down in the frustration over learning to mix and apply paints.

The Amiga's role

I was fascinated to see several of the artists who were starting out on the Amiga eager to get over to the "big machines" in the other room: the Macs and the Targa boards. After a couple of days, many came trickling back into the Amiga room and booted DPaint III. They had liked the resolution, the color, the stable image of the other systems, but the intuitive nature of the Amiga and software like DPaint was compelling. Again and again I heard the same thing, "the Amiga is more fun."

One specific advantage the Amiga has over other systems is its facility with animation. This became the main focus of many of the students. Although beautiful images were being produced in the Mac/Targa room, toward the last week the Amigas were booked solid day and night. I heard one student grumble from the doorway, "Oh well, I guess I'll



Scary monsters and super creeps: scenes from "Stormy Night", an animation by Dean Wellins done in traditional cel animation style using DPaint III.

go back and work on the Everex for a while, but call me if an Amiga opens up." Animation was the "Amiga factor" above all else.

This campus is hopping

Steve Segal gave a great demonstration of traditional cel technique using DPaint. This was the direction most of the artists followed. He also showed Sculpt-4D to a fascinated crowd, but no one really pursued it. The immediacy of DPaint, Animator, The Director and Photon Paint seemed more alluring. Gene Brawn showed his rotoscoping methods in detail, and several students followed that lead with excellent results. I introduced a few students to The Director so they could incorporate special effects and titling into ANIM's created with DPaint and FrameGrabber.

With so much interest in animation, RAM became a critical issue. Four 2000's had 3 meg, and one had 7 meg, a 68030 and fat Agnus. The rest of the machines were 500's with one meg. These were fine for basic work and any painting, but they topped out pretty quickly on animations. Students resorted to producing animations in short segments to append together later on the larger machines. Memory conservation became a valued craft. Students soon learned to close the Workbench and avoid running background programs like virus checkers. We also had the advantage of good insert editing video equipment. Students could produce long animations in segments, as in the two pieces illustrated here, then assemble them at the video stage, adding a soundtrack. The memory issue was somewhat circumvented and, thanks to the skilled video coaching of Rob and Jody, the results were very professional.

The animation frames shown here demonstrate the variety of approaches students took in their work. "Stormy Night", by Dean Wellins, was done in traditional cel animation style using DPaint III. Each segment was first rendered as line drawings, then colored with the FILL tool after checking continuity. The piece runs remarkably smoothly since Dean had the skill and patience to design the piece for a 15 frame-per-second play rate. This requires subtle incremental changes frame by frame

The Amiga In Higher Education

When it comes to college life, the Amiga has not yet enjoyed the acceptance other computers have. Rutgers University, in New Jersey, is one school that has recognized the special value of the Amiga. The Amiga Users Group of South Jersey (SJAUG) held its June meeting in the Fine Arts Building on the Camden Campus of Rutgers University. SJAUG is now planning to hold its regularly scheduled meetings at Rutgers. Those interested in attending the meetings can call or write the SJAUG. (See the phone number and address below.)

The June meeting began just after 7 pm with the usual organizational items, but the meeting concluded with a presentation by Professor Maria Palazzi of the Rutgers Fine Arts Department. Professor Palazzi and Rutgers University have put together quite a powerful setup for graphic arts students! The one-year-old lab houses six A2000s, each with 3 megabytes of memory. Each 2000 is networked to a Sun 3/60 workstation through an Ameristar Ethernet card. The Sun has two 141 megabyte disk drives from which the students are assigned workspace. The lab is used by the Graphic Arts curriculum, but students from the Computer Science department are beginning to show an interest in doing graphics-related programming on the Amiga. Professor Palazzi mentioned that students fill both drives twice a semester. (Sounds like the Prof works them pretty hard!) The students can use files on the Sun the same as if they each had dedicated hard drives!

A few of Professor Palazzi's students created some very interesting animations which were shown on a videotape made by the class. One of the assigned projects involved the image of a teapot. The students were required to create an animation starting and ending with the same image. Some of the final animations were included on the videotape. Professor Palazzi commented that while the Sun has considerable graphics capabilities, it is not as friendly as the Amiga! Professor Palazzi would like to get 68020 accelerator cards for each A2000, the reason being speed. The simple image of a teapot took about 35 seconds to render under Sculpt 3D. An identical rendering on a A2500 with a 68020 took just 15 seconds! Since much of the student work requires a lot of complex rendering, a good amount of time could be saved. The images being rendered were complex enough that students would set up their image, start rendering and leave it running all night!

Professor Palazzi hasn't had an easy time getting the Amiga into Rutgers. She attempted to get some help from Commodore in setting up the graphics lab. Commodore showed little interest. If those guys in West Chester could only see what has been done at Rutgers. The marketing types at Commodore don't realize how many systems a setup like this would sell! The students that go through that kind of a curriculum will most likely buy an Amiga since they are accustomed to its operation. Can't they understand why companies like Apple give big student discounts? Speaking of Apple, there is a Mac II in the lab. When a student helping with the demonstrations was asked what it is used for, the reply was, "Desktop publishing stuff, not much else." The student definitely showed a bias towards the Amiga!

Next, Philip Imbrenda, of TV One (TV One creates commercials using the Amiga) showed off the A2500. Phil's company does a lot of commercials on the Amiga, and Phil is going to head the special interest video group. He exhibited several examples of his work and created some new animations for all to see. One important point he made was that, while the Amiga may not have the same video quality when compared to some of the "professional" systems, those systems can cost $50,000 to $100,000. When comparing the two, the Amiga fills the low-cost video system market and delivers comparable quality.

Much of the meeting was dedicated to video. Most of the 50 attendees showed a strong interest in video. It was kind of nice to meet some of the folks I had spoken with on local bulletin boards, matching faces with names! I am certain that the Amiga Users Group of South Jersey will be a success.

For additional information on AUG, call (609) 667-2526 or write them at: AUG, P.O. Box 3761, Cherry Hill, NJ 08034. —Tony Preston

and dedication on the artist's part to create so many drawings for each second of play. The results are clean and professional, justifying the extra effort.

To keep the ANIM size as small as possible and to make efficient use of available RAM, good palette management is important. If 16 colors or even 8 are sufficient, then ANIM size is significantly reduced. Animations with large, clean areas of color translate well to video, and it is often possible to design beautiful pieces around an eight-color palette.

When transferring to video, color choice is crucial. Many colors which look great in RGB simply do not work in NTSC. Red is notorious for "crawling" into other colors when transferred to NTSC. A good rule of thumb in video palettes is to use no color with an R, G, or B VALUE higher than 12. Very dark colors tend to be indistinguishable from each other.

The opening scene of "Stormy Night" makes clever use of DPaint's MOVE requester to achieve a "multi-plane" effect as the viewer zooms in on the house through the branches of a tree in the storm. The house was created full screen, then pushed back on the Z axis in the MOVE requester and moved toward the screen plane brush position in the same total number of frames as the ANIM. The leaf brushes were then overlaid. During the zoom, the leaves were doing X and Y transitions to give the multi-plane effect poineered by Disney in his early features.

"The Hit", by Neal Stillman, demonstrates a visual style unusual for a computer animation. The use of black & white, and the dark, soft-focus, film-noire look are very effective. The painting technique used in the animation is a monochrome technique that I teach. It relies on having a sequential gradation of values from dark to light in the palette. With this entire sequence selected in DPaint as a RANGE, the powerful BLEND and SHADE tools can operate. In SHADE, one mouse button lightens and the other darkens. BLEND pushes areas of the image in the direction of brush movement, creating a soft edge as it goes. Neal also made extensive use of the SMOOTH mode to soften and blur portions of each final image frame and get away from the crisp, pixel look of typical lo-res animation.

This is a very stylish piece, and a great example of effective use of a limited technique. Neal learned the fundamental technique in ten minutes and stuck with it through the entire project. It is not always essential to know the software inside-out to achieve interesting results. I feel it is quite easy to get a student going on a good project quickly with a few basic tools, bringing the rest of the functions in little by little. In my opinion, the best software lends itself to this approach.

One valuable tip I might pass along to anyone creating animations is to save under two names. While working on an ANIM, updating it and diligently saving every few minutes still leaves the project vulnerable to Read-Write Error Guru crashes. The tragic thing is that such a Guru in the middle of a save not only means a trashed ANIM file on the disk, but a system lockup that loses the only other copy of the ANIM, the one currently active in the paint program. For safety, save duplicate copies of the ANIM under different file names, then save to them alternately as work progresses. In addition, important project disks should always be backed up. Electronic art is a deceptively tenuous medium.

While all this animation was going on in the Amiga room, and beautiful high-resolution images were coming out of the Mac/Targa lab, Rob and Jody had an amazing performance studio set up in the adjoining room. Students were creating images and brushes in the Amiga room and using them with the Mandala performance software next door. They were able to create visual "hot-spots" on the screen which could be "touched" by the performer as their live image was genlocked through the Amiga. Keyed to sound and image events, intricate real-time performances were created. Dancers were drafted from other classes for collaborations, and the studio was packed with onlookers as they whirled under the hot lights triggering chimes and thunder from the large amps with each kick and sweep.

Some of us used the performance studio for relaxation. I recall a beautiful time around two in the morning when Shu-Lin, a tireless animation student from Taiwan was well into 40 hours without sleep. I began hearing clear, structured music from the performance room, getting cleaner and cleaner as time passed. This went on for over half an hour. Looking into the dark room, I could see Shu-Lin lost in tranquility under the video light, her arms moving in smooth patterns while she stood relaxed, fixed on her own feedback.

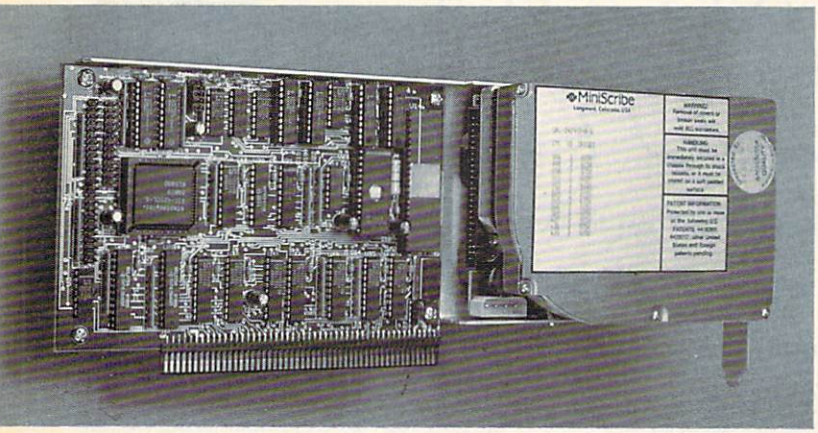
I looked around the Amiga room at the dozen or so students still working at this late hour. Gene was showing a sophisticated digitizing technique to a small group in one corner while others were locked into the mental groove of frame-by-frame animation. Some were just staring into space, lost in Shu-Lin's genlocked mantra. It occurred to me what a quiet pleasure it was to see learning happening as it should.

•AC•

For Information Contact:
CSU Summer Arts
The California State University
Office of the Chancellor
400 Golden Shore
Long Beach, CA 90802-4275

HardFrame/2000

The *Super-Speed*, DMA, SCSI Hard Disk Interface for the Amiga®2000



- **AutoBoots AmigaDOS 1.3 *(Price Includes HardFrame Eprom!)***
- ***Directly* Boots the New Fast-File System! *(Doesn't Need Old FS!)***
- **Auto-mounts All Hard Disk Partitions *(no Mount List Required!)***
- **Designed-in, Ultra Strong, Multitasking Performance**
- **High Quality Metal Frame for Stable, On-Card, Hard Disk Mounting**
- **Power Cabling Directly from Card to Disk**
- **50-pin Cable Included**
- **Supports up to *seven* SCSI hard disks of any size**

How fast is *fast*? **HardFrame/2000** transfers data at Amiga bus speeds! It's actually faster than the hard disk mechanism itself! And even more important in the Amiga's multitasking environment, **HardFrame/2000** has extremely efficient DMA circuitry to get on and off the bus in almost no time at all: 280ns to get on; 200ns to get off. And it's true, dedicated DMA, too! **HardFrame/2000** autoboots *and* automounts *directly* into the AmigaDOS™ 1.3 Fast File System (old file system partitions are *not* needed!). The core of any DMA SCSI interface is in its SCSI protocol chip and DMA chip. MicroBotics has chosen the new, high performance Adaptec AIC-6250 SCSI chip, capable of up to 5 megabytes per second raw transfer speed, and the Signetics 68430 DMA chip running at 12.5 megahertz. Then we added additional FIFO buffering and enabled 16-bit wide data transfers for maximum throughput. The sophisticated design of **HardFrame/2000** provides for automatic SCSI arbitration, selection and reselection. The hardware supports either synchronous or asynchronous data transfer. **HardFrame/2000** can function as either the SCSI bus initiator or the target and can reside in a multiple master environment. Physically, **HardFrame/2000** is optimally flexible: the compact, half-size card comes attached to a full length, plated aluminum frame. The frame has mounting holes positioned to accept standard, 3.5" SCSI hard disk units such as those manufactured by MiniScribe, Seagate, Rodime, and others (hard disk mechanisms must be supplied by the user or his dealer as a separate purchase item). Alternatively, you can cable-connect to a SCSI drive mounted in your Amiga's disk bay or in an external chassis. As many as seven hard disks may be connected to a single **HardFrame/2000**. There is no size limit on each disk. **HardFrame/2000** includes a 50-pin SCSI cable and header connectors for either 50-pin or 25-pin cable connection. Also included is a current tap to power frame-mounted drives directly from the slot itself. **HardFrame/2000** comes complete with driver, installation, and diagnostic software.

Available NOW! Suggested list price, $329 (hard disk not included)
Frameless version: $299.00. See your Amiga Dealer.

The HardFrame/2000 photo shows the product with a MiniScribe twenty megabyte hard disk installed. Hard disks are *not* included in the purchase price of HardFrame. Note that if placed in the first slot, HardFrame uses only one slot even with a disk attached.

New!
8-UP! (DIP) FastRAM

Another great memory board from MicroBotics, 8-UP! (DIP) is the "brother" of the original 8-UP! (which uses SIMMs and PopSIMMs to fill its memory space). 8-UP! (DIP) uses conventional 1 megabit RAM chips in standard sockets to provide your Amiga 2000 with 2, 4, 6, or 8 megabytes of autoconfiguring FastRAM! 8-UP! (DIP) is a super efficient CMOS design for lowpower consumption and high reliability. Suggested list price, $239 (0k installed)

MicroBotics, Inc.
Great Products Since the Amiga Was Born!
811 Alpha Drive, Suite 335, Richardson, Texas 75081 (214) 437-5330

Join MicroBotics ONLINE TECHNICAL SUPPORT CONFERENCE ON BIX (The Byte Information Exchange) –call 1-800-227-2983 for BIX membership information!

Tell your dealer he can quick-order from MicroBotics directly – no minimum quantity –show him this ad!

READING, 'RITING, AND RESOLUTION

A look at three Amiga graphics programs designed with kids in mind

by Joe DiCara

This article will review three paint programs primarily designed for preschool and elementary grades. Each program will be explored to learn its capabilities, features and usefulness. The use of these programs as stepping stones to introduce both children and adults to the world of computers and computer graphics will then be covered.

In The Beginning

The promise held out by the Amiga to education has always been great. Its versatility and friendly interface, as well as its graphics and sound capabilities make it a natural for school and home use. But as we have learned since the Amiga's introduction, having the right hardware parts does not guarantee immediate success. The key catalyst, of course, is software—the mightiest engine is nothing more than a boat anchor without the proper fuel.

Early on, some folks saw this computer's educational potential and quickly began creating applications to meet the need, but a year after the machine's introduction there were few programs of this type available. Most programs were ports from other machines—text only, written in BASIC, and utilizing none of the Amiga's advanced features. For those few that made the extra effort to use the speech, sound, and special graphics abilities, such as HAM (hold and modify), their programs found a very limited market. Part of the responsibility for this falls upon Commodore's decision, conscience or otherwise, to stand back from the educational market. With the absence of Commodore's leverage to get these new computers through the front door of schools, these early pioneers were, in effect, left holding the bag.

Fortunately for all of us, Electronic Arts and Dan Silva also realized this machine's promise and capabilities, and created DeluxePaint. Here was a program that everyone could use for enjoyment, education or productivity. Soon others followed, some concentrating their efforts toward special needs, namely children. With these new tools, the Amiga managed to hold on and wedge its way into industry and education.

Talking Coloring Book

The first program we will examine, Talking Coloring Book, was first introduced in early 1986 by JMH Software. This program is aimed straight at preschool children. While it does not use high-powered graphics (only nine crayons are available) it does take advantage of the mouse interface and most importantly, it uses the Amiga's sound capability to produce good clean understandable speech. JMH has provided a very enjoyable and useful program that not only introduces basic art concepts but also teaches relationships between the spoken and written word.

Upon boot-up, a colorful clown introduces the program. Clicking on the continue box brings up the options menu. At this point, a voice asks the child to make a selection. The four choices—Demonstration, Practice, Color, and Draw—can be selected either by a click of the mouse or typing the proper letter on the keyboard.

The Demonstration selection teaches the student the relationship between a color word and the associated color. This sequence can be repeated, or a new color can be displayed simply by moving the mouse pointer to the proper box.

The Practice option verbally directs the user to identify each of the nine colors. Here again, the mouse pointer is moved and clicked to select the color. The computer patiently teaches the student, praising correct choices or politely offering two more chances to pick the correct color.

The third option, Color, loads a menu of permanent pictures that may be colored. These are also selectable either by mouse or keyboard. Upon selection, an uncolored picture is quickly loaded. Off to one side are the crayons. A nice feature here is that the arrangement of the crayons is random. Each time a picture loads, the crayons will be in a different order, thereby assuring that the user is learning the color, not just its position on the screen. The student selects a crayon, the crayon's color is turned on, and the computer says the name of the color to reinforce the learning process. The pointer is then moved to the region to be colored, and a click of the mouse flood-fills it with the selected color. An eraser is available if the young artist changes his or her mind.

The final option, Draw, allows new pictures to be drawn, changed and stored. After saving a drawing, these pictures are available in the Color menu for coloring.

MyPaint

MyPaint, released in 1989 by Prism Computer Products, is a true paint program designed with kids in mind. The

form and format of this program resembles professional paint programs. The first of the paint tools available is the paint color box. When a color is selected a smiling face appears in that color to verify and remind the user of the color chosen from the palette of twelve colors. The next tool is Paint Brushes. The user has a choice of a wide or narrow brush. The last tool in this group, Fill 'Em Up, flood-fills the drawing.

A number of paint effect tools is provided. The first is Flashing Colors. Selecting this icon color-cycles all colors except white, black, gray, flesh and brown. The Mirroring tool duplicates the image being drawn in each quadrant of the screen. The final tool in this group is Color Brush. Selecting this option will produce a multi-colored brush. All colors are used in this effect, except those mentioned in Flashing Colors.

Other special features and capabilities include the ability to choose and load one of the 28 drawings, a clear screen function, a gadget called Surprise Picture that randomly selects any of the stored images, and Listen to Sounds. This last function will produce an appropriate sound the first time a picture is selected in a session.

MyPaint is a well thought-out program. Everything from the yellow diskette to the animated icons were designed with children in mind. It is simple to use, yet provides all the tools that will allow a young creative mind to expand.

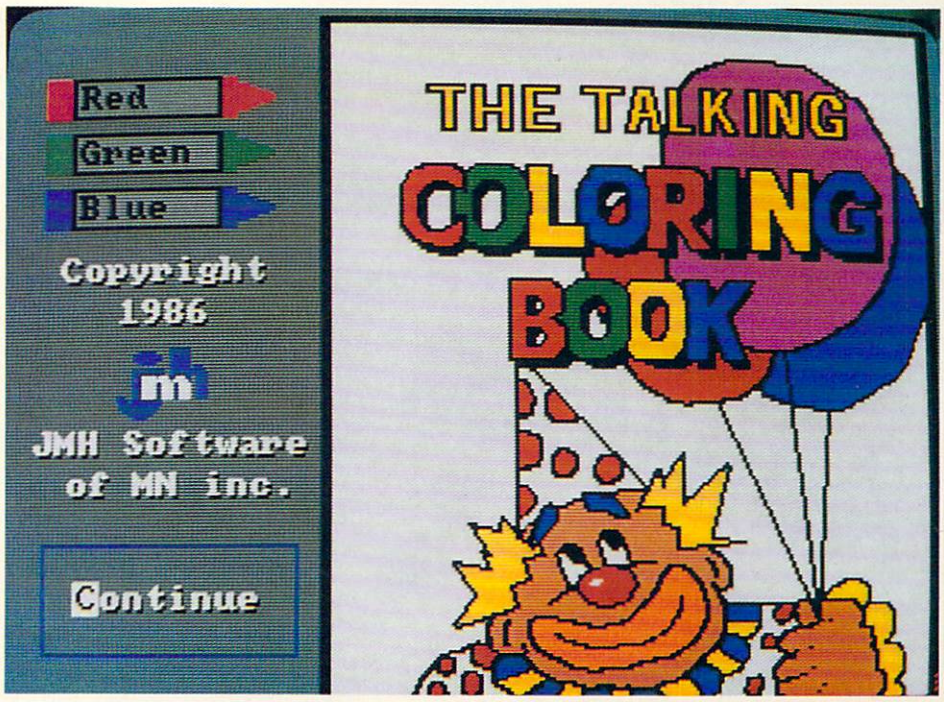
Talking Animator

Talking Animator, released in 1989 by JMH Software, combines many of the features provided in the two previously discussed programs, then adds text, speech, a palette of 8 colors and, as the title implies, the ability to do simple animation. In the documentation, JMH calls this program an idea presenter—and rightly so. Talking Animator is more than a paint program. With the ability to print any image or to interface to videotape, via a genlock, the user can produce a simple greeting note or perhaps graphically portray what their summer was like.

Though Talking Animator's drawing palette is limited to 8 colors, a requester can be called up, allowing a color to be changed to any one of the Amiga's 4096 available colors. While the current page is limited to the eight selected colors, every page within the animation can have its own custom palette.

Drawing tools are limited to just two brush sizes—straightline draw, and color flood-fill. Text can be typed in the picture simply by positioning the cursor where desired and typing. When this page is displayed in the animation, the text on screen is spoken. Be careful here, as this feature uses the Amiga narrator. Some words, though typed correctly, will not sound right due to the phonetic interpretation of the text. To help, a Talk gadget can be clicked on at any time, allowing the word to be corrected immediately.

Perhaps of the ten gadgets available, the Copy and Flip functions deserve the most explanation. After the first page is drawn, a click on the Copy gadget produces a duplicate picture and the page counter is increased. Now the next page of the animation is drawn. A nice animation aid is available here. By holding the "control" key down and clicking on the Page gadget, the previous drawing is revealed by ghosted blue lines. Now the current page can be drawn with easy reference to any existing page—ahead or behind. It's a simple but very effective technique. To



Hey, got any cotton candy?: the opening screen of the Talking Coloring Book by JMH Software

preview the animation, just click on Flip and behold the creation. Remember to use the text capability; it adds spoken narration that greatly enhances the picture-story with a fun new area of expression. Only the amount of available memory will limit the size of the animation.

Included with the main program disk is another disk containing demonstration animations, each requires at least one megabyte of memory. Also included is a single page of documentation. This is supplemented by on-line, spoken instructions for all functions.

The Sum is greater than the parts

As stated, I would like to explain how these three programs can be used to introduce and develop anyone's interest in computers and, in particular, prepare them for the more advanced paint/animattion programs now available to the Amiga community.

Talking Coloring Book is an ideal starting place for anyone, especially preschoolers. On the surface, it occupies the child with a very interesting succession of "fill-in-the-color" drawings. But while the child is exploring how a green bear might look, they are quietly learning to interface with the computer. They quickly develop the hand/eye coordination necessary to operate the computer, mouse, and program while creating and painting a picture.

MyPaint builds nicely on the foundation laid by Talking Coloring Book. The program is specifically designed to introduce the icon-style of interface. The animated icon's large size and color intuitively instruct on how to use this program, resulting in the student quickly accessing and using all of its functions. For the budding art student, the learning curve is also smooth and logical. Talking Coloring Book introduces the basics of color composition through the use of pen and ink-type drawings, then with MyPaint, all the basic tools that enable students to create more advanced computer art are provided.

Talking Animator brings all the tools, techniques, and the exposure of a sophisticated user interface together, while providing the new tools and environment that will allow the student to begin creating and producing useful computer art.

More important, Talking Animator is not the end of the line for a young artist or graphic designer. The new tools, techniques, and concepts introduced by this program open the door to really high-power paint/animation programs. Power paint programs such as DeluxePaint and Photon Paint can be used immediately and without intimidation. The concepts of programs such as PAGEflipper Plus F/X by Mindware International, Fantavision by Broderbund Software, and the like will not be lost or unfamiliar to anyone who has learned to use these three entry-level programs.

All three of these programs are excellent and very effective for the targeted user. Hopefully they will now be used in a structured and organized manner, so their combined power and teaching ability may be realized.

The far-sighted developers and programmers of these and other educational packages are owed a great amount of gratitude and appreciation. For without their efforts, Amigas would be nothing more than expensive paper weights. I would hope this point is not lost on the powers that be at Commodore International.

•AC•

Loud clothes seem to be the vogue: yet another clown, this time in MyPaint by Prism Computer Products

Opening screen to the Talking Animator by JMH Software

Products Mentioned

Talking Coloring Book
JMH Software of Minnesota, Inc.
7200 Hemlock, LN.
Maple Grove, MN 55369
(612) 424-5464
Price $29.95
Inquiry #206

Talking Animator
JMH Software of Minnesota, Inc.
Price $49.95
Inquiry #207

MyPaint
Prism Computer Products
Distributed by Centaur Software
P.O. Box 4500
Redondo Beach, CA 90278
(213) 542-2226
Inquiry #208

Commodore's Educator!

CBM uses video to reach the classroom.

Commodore's "Amiga — Creativity in the Classroom" is a new sixteen-minute video demonstrating the Amiga's extreme flexibility by showcasing teachers and students using the Amiga in the classroom and beyond. From an elementary art class in Columbia, South Carolina to Art Education at Ohio State University, teachers and students demonstrate their creative talents through the Amiga. Educators, filmed against a background of busy students and projects, speak freely about the Amiga's capability and their use of the machine.

In quick succession, the Amiga is demonstrated by teachers, students, and their projects as an administrative computer, an artist's medium, a super MS-DOS clone, a musical instrument, a sound editor, a desktop publishing machine, and even the heart of a television studio. Between shots of Amiga generated graphics and sound, the Amiga is seen in both large and small computer labs. Students are shown performing their own computer presentations and videos.

> "We believe that it (the video) will help educators realize the **amazing** capabilities of the Amiga..."
> John DiLullo
> Commodore's Education Manager

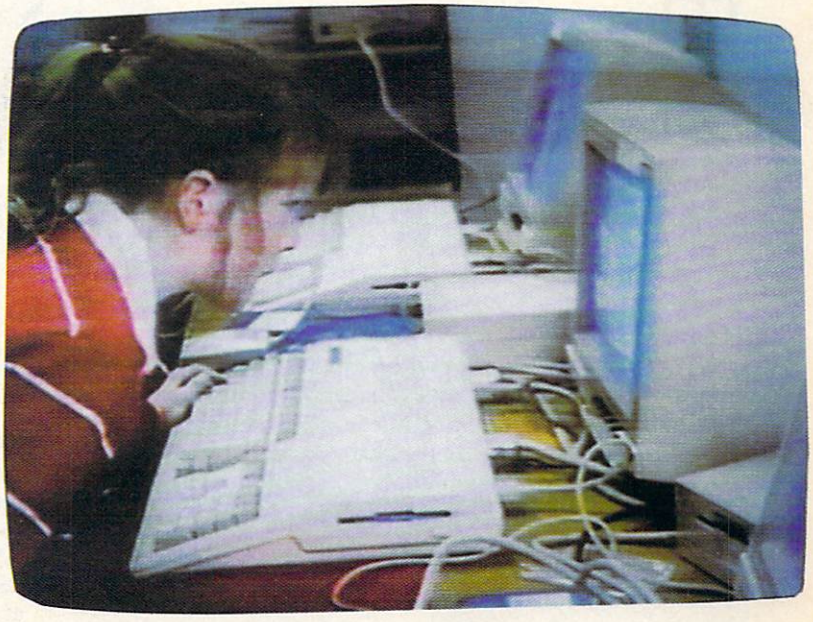
In a press release from Commodore, Mr. John DiLullo, Commodore's Education Manager, stated, "The new Amiga Video demonstrates some of the many ways that the Amiga can be used effectively in the classroom. We believe that it (the video) will help educators realize the amazing capabilities of the Amiga, and how these capabilities can best be utilized in improving instruction."

Little is said throughout the video about the Amiga's programming environment or languages. The video presents the Amiga without involving technical aspects of the machine. The Amiga is presented as a solution to teaching concerns and as a tool in creative activities.

For a copy of "Amiga — Creativity in the Classroom" please write to:

John DiLullo
Education Manager
Commodore Business Machines, Inc.
1200 Wilson Drive
West Chester, PA 19380

Inquiry #222

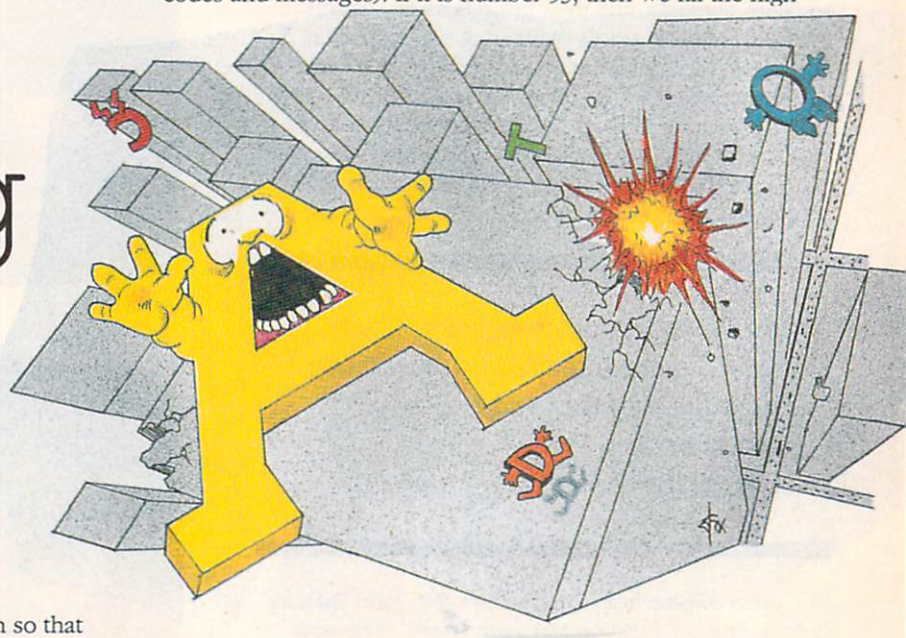


Scenes from a video

From placing the Amiga on a pedestal to showing the excitement of achievement and discovery in a child's eyes, Commodore's new video presents the Amiga as an indispensable tool for education.

Typing Tutor

by Michael "Chip" Morrison



The city of Keycaps is being threatened by a first strike of capital letters from the country of Thesaurus. They have even gone so far as to use the NATO-outlawed weapon of ultimate destruction. That's right: vowels! It is now up to you to fend off this unprovoked, hostile attack. You are armed with only one weapon. It is the weapon of choice. It is MAK (My Amiga Keyboard). No time for nonsense, you must move forth and destroy this sinister enemy!

Method to the madness

As you may (or may not) have guessed, this game is an elementary typing tutorial. I had three reasons for writing this program:

1) I wanted to write a program that uses only the standard AmigaBASIC commands. No fancy library calls here; you can write very nice programs with the commands available within AmigaBASIC.

2) I wanted the source code to be small enough so that typing it in could be accomplished within the normal life span of an Amigan.

3) I wanted to show how you can write easy-to-read code using descriptive variable names and labels. For example, if you have a routine that updates the screen, you could use a label named "us:" or "this.is.the.routine.that.updates.the.screen". Well, maybe not that descriptive.

The game

In Typing Tutor, letters fall from the sky at random locations. Before the letter falls too far, you must type the corresponding letter on your keyboard to save the skyscraper below from destruction. The letters start off falling at a reasonable speed. As the game progresses, the letters fall faster and faster.

Eventually, you will not be able to react quickly enough. You will then be subject to the desperate screams of several thousand people being crushed to death by the falling letter that you so irresponsibly failed to locate on your keyboard (think of someone besides yourself for a change, will ya?)

Once you have let three letters fall, the game ends. If you score high enough, you will be prompted to enter your name in the high score list. This should be very simple the first 10 times you play because the program creates a blank high score list if it cannot find one in the current directory.

Program clarification

Typing Tutor is a fairly straightforward, no-octane AmigaBASIC program. However, there are a few pieces of the listing that may be somewhat confusing. In the init.hi.score section of code we use the ON ERROR GOTO command. This command allows us to trap certain errors instead of stopping the program and putting up an error message. Any time an error occurs in our program, it will go to the routine called "handle.error".

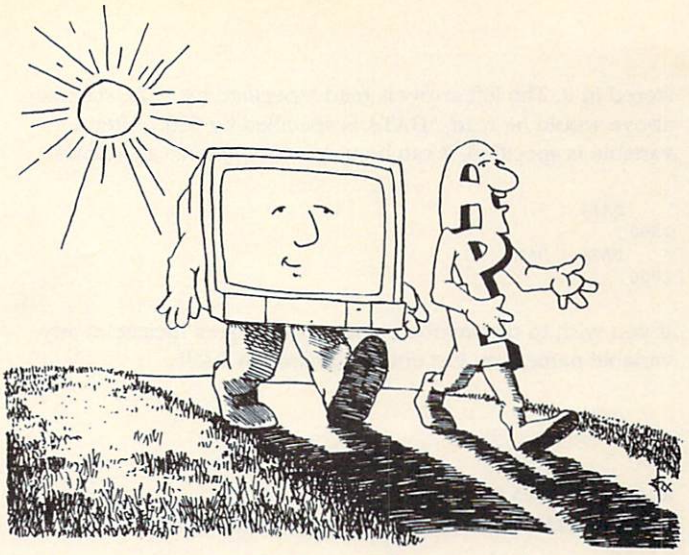
The program tries to open the high score file named "letters,.score". If it is not found in the current directory, an error is returned by the system. The handle.error code checks to see if the error number is 53. Error 53 is "FILE NOT FOUND" (see AmigaBASIC, Appendix B for a complete listing of error codes and messages). If it is number 53, then we fill the high score table with "Empty for now". If it is another error, we use the command ON ERROR GOTO 0. This lets the system handle any errors.

The "get.char" routine could use some explaining. In this routine, the letter to be dropped and the location it is to be dropped in is randomly selected. The same set of functions are used to generate both random numbers. Way back at the beginning of the program, under the "init.some.stuff" label, the RANDOMIZE TIMER command is used. This command insures that the function RND() will generate different random numbers each time the game is played. RND(1) returns a number greater than 0 and less than 1. To generate a number between 1 and 26 (the alphabet), we first multiply by 26 and then use the INT command to get rid of the decimal part.

Next, we add 65 to that number to get an ASCII value from A to Z (see AmigaBASIC, Appendix A: Character Codes for a list of all ASCII codes). After all this, we use the CHR$() function to change our number into a valid string. We then enter a WHILE/WEND loop. Inside the WHILE/WEND loop is a FOR/NEXT loop which simply causes a delay. This delay is

(continued on page 52)

APL and the Amiga:
A Friendly Pair



by Henry T. Lippert, Ed.D.

Most *Amazing* readers now know the excitement of having, right on their own desks, one of the most spectacular computers ever produced—the Amiga. One by one, its outstanding hardware features are being revealed by the large number of programs that are now available. From the start, the Amiga had several languages available in which to write programs. The C language, of course, was the most powerful, while BASIC was the most popular. Another language has appeared on the scene, going unnoticed by nearly everyone. Hidden away from the public eye, this computer language, called APL, is like no other. (An excellent review of APL—"APL.68000" by Roger Nelson—appeared in *AC*, V3.5, p. 49.)

It is difficult to work up much enthusiasm for most of the common computer languages. Languages are expected to be there in order for the computer to be respectable in the marketplace. It may even seem strange to raise the issue of a computer language being enjoyable. Languages simply aren't a whole lot of fun. Nevertheless, it is a common experience for persons to be very excited when their first non-trivial computer program actually works. It is just as common for the seasoned computer programmer to have programming excitement that parallels that associated with other activities—such as cleaning the gerbil cage or watching the trees grow. The initial excitement about a language gives way to the satisfaction of having accomplished a difficult task, having created a "new feel" or a best-selling program, or perhaps having received a royalty or salary check.

The excitement and fun returns when using APL to drive a computer. APL stands for "A Programming Language." No kidding. The name came from a mathematician, Dr. Kenneth E. Iverson. The story goes that, when asked what he was working on, he replied, "Oh, just a programming language." The name stuck, and the language became one of the greatest enigmas in all computerdom.

People in the computing world either dislike (read dismiss, despise, or hate) APL or they love it. Most of those who do not like APL have seen some code but know little about it. Most of those who love it are really in love. The true APL-type person believes that APL is the programming language and that it will be in wide use some day. APL is used extensively to drive mainframe computers, but it is nearly always hidden from view. Some people seem to be embarrassed to admit they know that APL even exists. For example, at the Thomas J. Watson Research Laboratory, IBM hid APL from view for many years before venturing an admission that they used it extensively as a language for scientific development, numerical analysis, internal operations, and data processing for the corporation itself. Most folks think that because APL looks nothing like all the other languages, it cannot really be a serious language. Well, it does look a little unusual.

Why bother with APL, you say? Let's take a look at some of the reasons. First, APL notation is essentially the same as that used for plain old familiar mathematics. Here are some APL statements:

```
2 + 6

473 x 8492

468 - 74

367.93 - 42.8
```

They look just like ordinary mathematical statements, right? In some sense, they are. APL uses the language of mathematics as much as possible. It is easy to see that you can begin to use APL without having to learn a computer language at all.

When statements such as those above are typed into the APL Interpreter, they are evaluated and an answer is displayed on the screen. The user of APL knows that APL is ready to accept a statement when the cursor appears indented seven spaces on the line. APL answers back at the left margin. The following gives you an idea of what a real APL session might look like:

```
      2 + 6
8
      12 x 12
144
```

These statements have no lasting effect. They are executed and the answer is displayed, but nothing else is done. You say, "Hey, that's just like a desk calculator!" Yes, you are correct; APL performs superbly as a desk calculator.

APL does more than that, too. It provides for storing statements for the long term. For example, the following statement:

```
DATA ← 500
```

sets up a storage area called DATA which has the number 500

stored in it. The left arrow is read "specified by." The statement above would be read, "DATA is specified by 500." After a variable is specified, it can be used in statements as follows:

```
    DATA x 3
1500
    DATA + DATA
1000
```

If you wish to display the contents of the area located at any variable name, you just enter the name by itself:

```
    DATA
500
```

and APL shows what is stored at the name.

The DATA could also be used in the following way:

```
    ANSWER ← DATA x DATA
    ANSWER
250000
```

where the first line would be read, "ANSWER is specified by DATA times DATA." Since the sentence began with a specification (left arrow), APL did not show the answer; it simply performed the operation. The second line, with ANSWER alone, told APL to show the contents of the variable ANSWER. APL replied with 250000 on the next line at the left margin.

The two statements could have been combined as the following statement:

```
    ⎕ ← ANSWER ← DATA x DATA
250000
```

The box (⎕) is symbolic of the display screen and told APL to show the contents of ANSWER. Of course, the answer was stored at ANSWER before its display. The statement would be read, "The display is specified by ANSWER, which is specified by DATA times DATA."

APL uses the four fundamental mathematical operators (+, -, x, and ÷) in their native forms. Because exponents are difficult to handle for the typical keyboard and many printers, one asterisk is used for indicating exponentiation. For example, the statement:

```
    DATA * 2
250000
```

yields the same answer as DATA x DATA, and is read, "DATA raised to the power of two."

The most important characteristic of APL which distinguishes it from other languages is that it is "array-oriented." For example, the following statement:

```
    DATA ← 2 4 6 7 5 8 12
```

writes over the previous value (500) and automatically makes the value stored at DATA into a 7-element vector of numbers. The entry of DATA alone now yields:

```
    DATA
2 4 6 7 5 8 12
```

showing that the new values have been stored at DATA. The following statement as an input:

```
    DATA x 6
12 24 36 42 30 48 72
```

and the result at the left margin show that the APL system took care of a lot of details for us. A legal operation—take the elements stored at DATA and multiply by six—was asked for, and APL performed it as requested. The following example illustrates again the concept of being array-oriented:

```
    DATA + DATA
4 8 12 14 10 16 24
```

This result shows that if APL is asked to add together two vectors of numbers, it does so element by element.

In most languages, a lot of bookkeeping is necessary to add the corresponding elements of two vectors, or to add a number to every element of a vector. APL keeps track of the size of vectors and performs arithmetic operations on them if requested to do so.

Before you think APL can do everything, try giving the following statement to APL for evaluation:

```
    DATA + 4 8 9
LENGTH ERROR
    DATA + 4 8 9
         ^
```

The resulting error message indicates that the 3-element vector of numbers, [4 8 9], is not the same length as DATA. How many elements does DATA have, anyway? In APL, finding vector lengths is simple. The following statement:

```
    ρDATA
7
```

indicates that DATA is a 7-element vector. Of course, APL cannot successfully add a 3-element vector to one of 7 elements.

Since storage areas in APL are dynamically allocated, rather than predefined as required by most languages, you may need to find out the size of an "object." The "rho" (ρ) requests the size of whatever immediately follows it. In the above example, there are 7 elements in the vector stored at DATA.

APL has a large number of already defined operations similar to rho. Dr. Iverson simply observed what programmers were spending their time doing. He noticed that in using languages such as FORTRAN and COBOL, the popular languages of the day, many operations were being coded repetitively every time a program was written. A great deal of time was spent on laborious indexing and "holding hands" with the computer, telling it over and over what to do. All of these things, he felt, could be left out of a programming language if the language provided for adequate description of what the desired output was to be like. All that was necessary was to emphasize the nature of the outcome expected of an operation, not how it was being done. Iverson simply defined such operations as a part of the language. For example, the slash (/) has the effect of extending a mathematical operation to elements of an array. It has the name "reduction" and operates as follows:

```
    +/ DATA
44
```

The extension of addition to the elements of the DATA array

placed a plus sign between each element of the array and reduced it to a single number (a scalar) by—yep, you guessed it—adding 'em up!

Take a more complex APL statement:

```
     (+/ DATA) ÷ ρDATA
6.285714286
```

which reads, "Sum reduce the elements at DATA and divide by the size of DATA." We might also have read the statement as, "Add up the numbers at DATA and divide by the number of elements stored at DATA." Hey, we just computed the arithmetic average of the numbers stored at DATA!

Well, readers, there is a lot more to APL. Whereas most languages have only the 5 arithmetic operations and the relational operators made available in branching and control statements, APL has some 60 to 80 operations that are defined as parts of the language. It was inevitable that APL would show up on the Amiga—a superior language on a superior computing engine. The combination of APL and the Amiga will no doubt place us in a permanent state of excitement.

Next time, we'll examine what it takes to write a program. All we did this time was to write a few statements, start to explore two of the new operators APL provides, store some data, analyze it, and show a little bit about the array orientation of APL. (Not bad for the small number of statements APL required to do all that!) The homework for all FORTRAN, BASIC, C, and COBOL programmers is to perform all the things accomplished here in APL and see how many statements (or key presses) it takes in those "easier" languages. The typical comparison has demonstrated a 10-to-1 advantage of APL over the traditional languages (FORTRAN and COBOL), using whatever measure you wish. Now you know why IBM has used it internally: it's efficient and profitable. Finally, the ultimate reasonably priced computer can now have the ultimate reasonably priced language—APL. They deserve each other.

•AC•

Dr. Henry T. Lippert is an educator. He has specialized in the application of computers in education and training and to the tasks of the instructor and the instructional designer. He was one of the original developers of Computer-Based Instruction (CBI) at the University of Illinois during the 1960's. Currently, he is Chief of the Instructional Methods Division at the Academy of Health Sciences in San Antonio, Texas. He has been responsible for the development and use of both microcomputers and dedicated CBI terminals for delivering instruction in medical technology to Army personnel.

APL Interpreter includes Reference Manuals in a hard slip cover, not copy-protected.

APL Interpreter for the Amiga:
Spencer Organization, Inc.
P. O. Box 248
Westwood, New Jersey 07675
(201) 666-6011
Price $99
Inquiry #205

BETTER TrackMouse

by Robert L. Katz

In the August 1988 issue of *Amazing Computing*, Darryl Joyce told readers how to convert an Atari Trackball into a "TrackMouse". This is a nice little modification. It is fairly easy to do, and the result is well worth the effort.

When I first saw one of these devices, I wondered why anyone would want one. After all, the Amiga comes with its own easy-to-use mouse. I knew about the advantages of track balls since I had used them during my military experience. I had even designed one into a command-and-control system we were building at work. But I had never seen their usefulness demonstrated with personal computers.

However, after talking to a few people in our user group, I realized that not everyone is fond of mice. For one thing, some people don't have good eye-hand coordination. For another, mice can take up a lot of desk space. A number of us here in Albuquerque have made this modification. Those who do mostly graphics-type work love them. They're able to lean back, put their feet up with TrackMouse in lap, and work in an ultra relaxed fashion.

Generally speaking, I liked my mouse. However, I disliked having my keyboard elevated so much above lap level. I had found a nice computer desk with a pull-out keyboard shelf. However, there was not quite enough room on that shelf for a mouse to sit next to the keyboard. I used it there, but still I felt cramped. After seeing a TrackMouse in operation, I began to think that maybe I didn't have to suffer this particular limitation of a mouse.

Jerry Pournelle commented on just this problem in his Chaos Manor column in the August 1988 BYTE. Jerry kept losing his mouse under piles of paper. He turned to a trackball, and found that it overcame many of the difficulties he experienced with mice. For CAD and similar graphics-oriented work, a trackball is at least as good as a mouse. Control is smooth, and it's much easier to move the cursor across long distances. A trackball also takes minimum desk space (even less if you sit back with it in your lap).

It takes two hands...

Since it seemed that a trackball would be a very nice alternative to a mouse, I bought one and made the modification. It was then that I learned about the click-and-drag disadvantage of a trackball. While, in many ways, a trackball is superior to a mouse, it is almost impossible to do click-and-drag operations with a trackball. While the mouse is a one-handed effort, the trackball is two-handed. It's even worse with the TrackMouse because the buttons are in awkward, entirely separate locations. After all, the Atari trackball was designed for use with video games. I had previously purchased a Wico trackball for my IBM PC at work. Wico had tried to simulate click-and-drag operation, but it just didn't work properly.

The shape of the Atari trackball housing does not lend itself to one-hand computer usage. It does not have sufficient depth, it is too high, and too wide. Hence, there is no place to rest your hand. I realized that solving part of the problem would mean rotating the whole thing 90 degrees clockwise. This

Table One

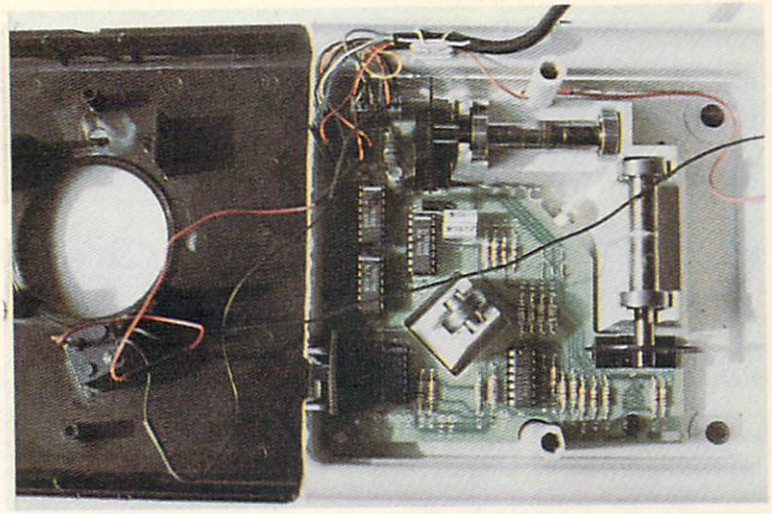
Original Header-to-Cable Connections:

HEADER PINS	CABLE PINS	COLORS
A	3	Green
B	4	Brown
C	1	White
D	2	Blue

New Connections:

HEADER PINS	CABLE PINS	COLORS
A	4	Brown
B	3	Green
C	2	Blue
D	1	White

places the short dimension under your hand, giving a nice palm rest. I further realized that there was no reason why the buttons could not be moved where I wanted them.



This photo shows the 'guts' of the trackmouse with the modifications made.

The one-hand trackball

So, after some thought and experimentation, I came up with a plan to make the trackball into a one-hand device. I first performed the modification illustrated in Darryl Joyce's article. I then further modified the trackball circuitry so I could turn the whole housing 90 degrees and still maintain proper ball rotation sensing. This is done in two steps:

First, switch the horizontal and vertical axes by switching four wires between the header and the new connecting cord. (See Table One for proper header-to-cable connections.)

The cable pins can be easily pulled out of the header and swapped around to the new configuration. This switches the horizontal and vertical axes and brings us to the second step:

The new horizontal axis rotation is backwards. To correct this, swap two circuit board leads. The photograph on page 3 of the modification article shows the circuit board clearly. Just to the left of the vertical trackball bearing rod are three traces going up, and then bending left by 45 degrees (see photo). Swap the outermost two of these three leads. Do this by cutting out about a 1/4 inch of each lead. Cut all the way through the trace at each end of a 1/4 inch section. Then use your knife to pull the section completely off the board.

This leaves enough room to solder the cross-connecting wires. Gently scrape the top of each lead, cleaning off the mask, until you have a smooth, shiny surface. Take two one-inch lengths of wire wrap and strip about a 1/4 inch of insulation from each end. Solder these wires in an X pattern across the two traces. The top left trace end goes to the bottom right trace end, and the top right trace end goes to the bottom left trace end. This completes the electrical modification.

Now we have to make the mechanical changes to give us a true TrackMouse. The left switch on the Amiga mouse is the actuate button. The leftmost button on the TrackMouse is the corresponding switch. Move it alongside the trackball by cutting off the top (old left) side of the trackball housing (which includes the switch). I used a Dremel Moto-Tool with a thin grinding wheel to make the cuts.

This leaves an opening in the housing that allows dust to get into the trackball mechanism. The opening must be closed off. This may be done in a variety of ways. I bought a sheet of 1/16-inch plastic from a local hobby store, and cut it to fit the opening. I then used Super Glue to fasten the plastic to the bottom half of the housing.

Since the old actuate switch requires too much pressure to activate, discard it, but keep the button. Cut out a hole in the top of the trackball housing about 3/4 of an inch away from the left edge of the ball. Epoxy a surplus keyboard switch to the underside of the housing surface so that the switch shaft sticks up through the hole. Then epoxy the Atari button to the switch shaft. Solder the old switch button wires to the new switch. To do this you need to extend the wires. Cut about four inches off the old connector cable to get the red and black wires that are needed. Follow the color coding to splice the new wires into the existing wires. After extending the switch wires, route them along the bottom of the case so they do not interfere with trackball operation. This completes the actuate button modification.

The right switch on the Amiga mouse is the select button. With the rotated housing, the previous right hand Atari button now falls under the right edge of my hand's heel. I decided to leave this button as is, but make it more convenient to operate. I positioned a stiff flat rubber strip across the bottom of the housing so its right end rests on the button. This looks very much like a keyboard space bar. I then epoxied the right bottom of the strip to the top of the button. Finally, I epoxied the left bottom of the strip to the housing.

Note that the Atari logo on the (old) bottom of the track ball housing is made of embossed foil and glued to the housing. It can be easily removed. I used 1/2 inch embossing label tape to make my own Amiga logo.

Putting it to Work

To use the TrackMouse, I rest my palm on the bottom edge of the housing. My fingers are free to position the trackball so I can select a desired menu item. When I rest my two middle fingers on the ball, my index finger falls naturally on the switch button.

With a little practice, click-and-drag operations are almost as easy as with the mouse. For other uses, I can easily move the ball with my middle fingers, and use my pointing finger to click on desired items. When I need to actuate the menus, I rotate my hand slightly to the right, and press down with the palm edge. The select switch is sufficiently stiff so that simply resting my palm on it does not actuate it accidentally. This modification has made the trackball so easy to use that I have permanently retired my mouse.

•AC•

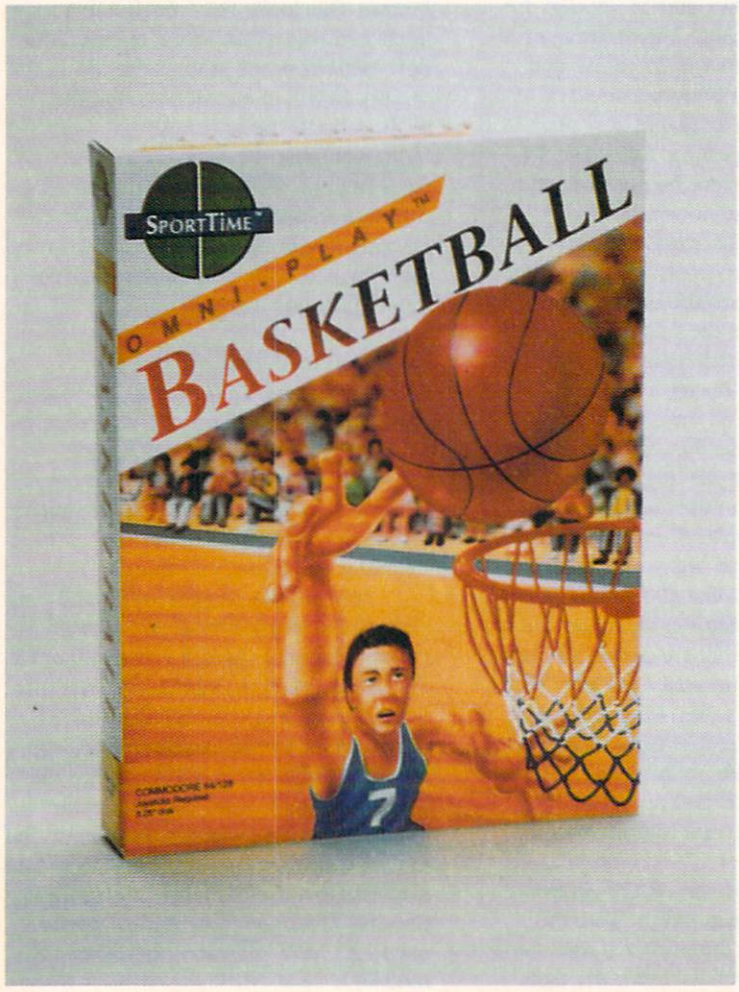
New Products and Other Neat Stuff

by Elizabeth G. Fedorzyn

Swish!

In your armchair basketball-star existence, do you find yourself torn between the desire to manage your own team and the overwhelming urge to run up the court with an oh-so-graceful lay-up. Well, SportTime may just have the basketball package for you. OMNI-PLAY Basketball, SportTime's latest release, offers every angle of basketball simulation-plus expandibility!

OMNI-PLAY lets you own, manage and coach your own basketball team. On the managerial level, you determine playoff structures, season lengths, etc. You can recruit new players from the minor leagues, trade players, or get your current herd of hoop jocks into shape at training camp. Players are not immune to the terrors of real life b-ball, either. Players age year after year, and suffer injuries as well (beware the possibility of a Celtics '88-'89-like club). You will be provided with complete stats on all 288 league players to help guide your decision-making as you work your way towards a championship and that ever-coveted SportTime trophy. And don't throw your jersey in the wash just yet. Because on the player's level, you're free to pass, shoot, rebound, slam it, make the J, and even shatter the backboard.



Every angle of hoop—plus expandibility: SportTime's OMNI-PLAY Basketball

What makes OMNI-PLAY Basketball unique is the game's expandibility feature. With the purchase of SportTime Option Modules and Support Disks, you can expand the peripherals of gameplay many fold. The "Side-View" Module, for example, lets you enjoy the game from a whole new camera perspective. With the "College League" Module, you can play with the teams, players, and stats from the 1989 college basketball season. Other Option Modules include the "Pro League", and the "Fantasy League", to name a few.

OMNI-PLAY even has its own answer to Bob Cousy. "The Nick and Bob Show" features SportTime's two favorite announcers, Nick "The Net" Jones and "Basket" Bob Smith. Before the start of each game and during halftime, The Net and Basket will provide you with their own analyses, predications, and general b-ball banter. (Any long periods of silence might be interpreted as The Net and Basket's having invited former President Reagan to do some commentary).

OMNI-PLAY Basketball requires a minimum 512K. Players can tip off against the computer, against one another, go two against the computer, or sit back and watch the computer chase itself up and down the boards. OMNI-PLAY Basketball—whether your idol is Pat Reilly or Pat Ewing, it's bound to be right up your court.

SportTime Computer Software
3187-G Airway Avenue
Costa Mesa, CA 92626
(714) 966-1311
OMNI-PLAY Basketball: $49.95
Inquiry #223

A contest, BASICally

Attention, True BASIC programmers! True BASIC Inc. has issued a call for entries in its First Annual Best-of BASIC Contest. True BASIC will award cash prizes to the most innovative code submitted in three different categories: Educational Software, Subroutine Libraries, and Other Applications. Cash prizes within each category are set at $250 for First Place, $150 for Second Place, and $100 for Third Place.

The panel of very distinguished judges will include Professor John G. Kemeny and Professor Thomas E. Kurtz, who together invented BASIC at Dartmouth College in 1964. Entries will be judged on the basis of their originality, usefulness, and programming style.

Contestants must include a copy of their True BASIC code on disk. The inclusion of documentation, on disk or in printed form, is encouraged. No entry fees are required. Entries must be received at True BASIC's West Lebanon, NH offices no later than 5:00 PM on November 17, 1989. Contest winners will be announced December 29, 1989.

For details regarding the First Annual Best-of-BASIC Contest, you may write True BASIC, Inc., 12 Commerce Drive, West Lebanon NH 03784, or call (603) 298-8517.

METCOM*89

The Mid-Cities Commodore Club will present METCOM*89 on Saturday, October 14. The show will feature presentations on various topics including "Computer Art" by Marilyn Coughran and "Commodore Update" by Andre Frech. The Mid-Cities Commodore Club will be sponsoring a number of activities including an Amiga workshop and a library of Amiga public domain software.

The show will be held at the Arlington Convention Center, 1200 Stadium Drive East, Arlington TX 76011. Showtime is from 10:00 AM to 5:00 PM. Admission is free. For more information, contact Ned Kelly at (817) 277-5825.

Playing God

Well, it took God six days took complete the task; let's see how long it takes the average Amigan to create a viable world. (Don't go scurrying to that bomb shelter just yet; it's only a game—and a rather provocative one at that).

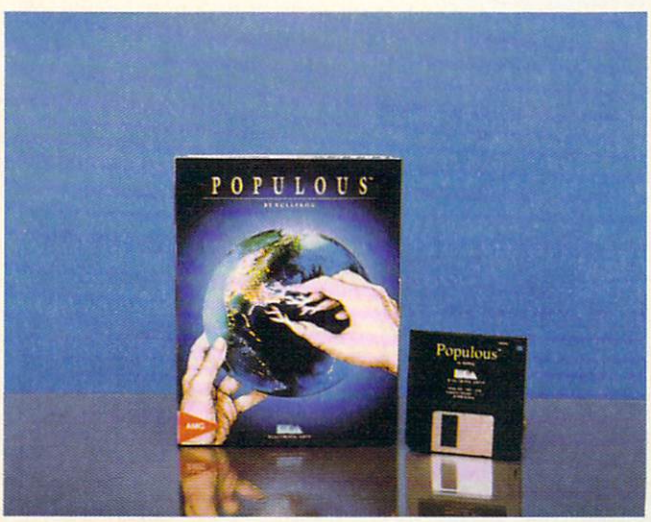
Populous, the hot new strategy/adventure game from Electonic Arts, places you in the role of a god (good or evil—your choice) who must build up a nation and, at the same time, destroy the quests of an opposing deity. In Populous, which was designed by the European artist group "Bullfrog", two nations are created, both with the need to populate, claim new territory, and attract new citizens. Both nations are also trying to increase their territory by spreading their population across the other's land.

Your initial goal in Populous is to promote growth in your nation so as to increase your power as a deity. You press across the land readying areas in which followers may build mud huts, houses, or castles and then settle. The larger the settlement, the more power you are given, since the more followers there are to worship you, the more power you are given to influence world affairs (sounds a little like MTV).

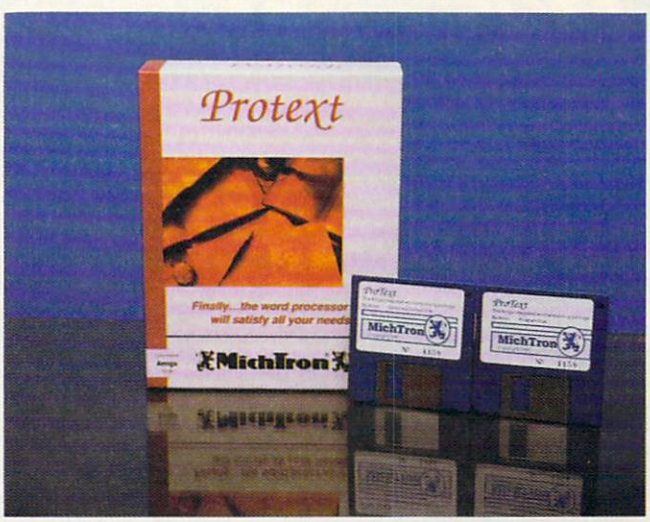
Of course, creating your own nation is not the greatest of your challenges. You also have your opponent's (un)godly ambitions to worry about. To ward off the opposing god and his plans for world control, you can use your power to turn into a Knight, then venture on a raid and destroy enemy buildings. But, heck, we're talking serious power here. Why not just put that amateurish raiding stuff aside and create a natural disaster. Nothing like a tornado or flood to really put a damper on an opposing deity's day. Both gods will also have to beware of nasty sea monsters and wizards who seek to wreak havoc on both nations. These preliminary battles will hopefully prepare you for the final conflict: Armageddon.

Populous offers an almost limitless amount of gameplay options, as players may venture into 128,000 worlds. The game provides two modes—a conquest mode and a custom mode. The conquest mode features hundreds of worlds based on four different terrains ranging from artic to desert landscapes with preset fighting conditions. The custom mode lets players design their own world and select conditions under which to populate and fight. You can play against the computer, or two players can wage war against one another via modem.

Populous features impressive 3D maps and animations, as well as digitized sound effects. According to Bullfrog's



The whole world in your hands: Populous from Electronic Arts



Word processor and then some: MichTron's ProText

head programmer, Peter Molyneux, "Populous incorporates the speed of an arcade game, the thought process of an adventure game, and the feel of a simulation. It is truly an interactive game that combines fun and strategy elements in one." Probably not too bad for the ol' ego either.

Populous
Electronic Arts
1820 Gateway Drive
San Mateo, CA 94404
(415) 571-7171
Price: $49.95
Inquiry #224

Give me a "C", a bouncy "C"

This month, Blue Ribbon Bakery will release Bars&Pipes, a breakthrough in MIDI music composition. With this first object-oriented music composition system, you needn't spend days (weeks, months) setting up your system. Nor do you need to take on a the persona of a eighteenth century German composer in order to compose music.

Bars&Pipes four main features include The Pipeline, The Toolbox, the Sequencer, and The Editor. Using The Pipeline, your musical input is guided from conception to performance. By arranging the pipes and valves, you direct the flow of musical information on a track-by-track basis. Each Pipeline can process information prior to or after recording. The Toolbox, features a host of different tools, including chord substitutor, event filter, and harmony generator, with which to process MIDI information as it flows through The Pipeline. And, if the available tools are still not enough to make you the next Bach, you can invent your own macro-tool with the program's Create-A-tool feature.

With Bars&Pipes' Sequencer, there is no limit to the number of tracks you can record. The Sequencer boasts, among other features, global cut, copy, and paste; auto-locate registers; looped mode recording; global display of music on all tracks; and rhythm, chord, key, lyric and time signature input. The Bars&Pipes Editor lets you view your music on a piano roll format, or as bars on a staff. You can open multiple edit windows at once. Using Tools, you can process sections on a note-by-note or phrase-by-phrase basis. Each note is heard as it is edited. You can also edit key, rhythm, and chord change information for algorithmic composition.

Bars and Pipes
Blue Ribbon Bakery
1248 Clairmont Road, Suite 3D
Atlanta, Georgia 30030
(404) 377-1514
Price: $250.00
Inquiry #225

All rolled into one

MichTron's new ProText is a fully integrated word processing package that combines the features of a word processor, text editor, and command line interpreter.

OTHER PRODUCTS RECEIVED:

Acft Pics
Tangent 270
2509 Dahlia
P.O. Box 38587
Denver, CO 80238
(303) 322-1262
Price: $49.95
Inquiry #227

AmigaDOS Reference Guide, 3rd edition
Chilton Book Company
One Chilton Way
Radnor, PA 19089
(215) 964-4000
Price: $21.95
Inquiry #228

Color Window 3.1
Kimbersoft
13281 72nd Avenue, Suite 201
Surrey, BC, Canada V3W 2N5
Inquiry #229

CrossDos
Consultron
11280 Parkview
Plymouth, MI 48170
(313) 459-7271
Price: $30.00
Inquiry #230

Denaris
Hard Wired Entertainment
2171 Dunwin Drive, Unit 13
Mississauga, Ontario, Canada, L5L 1X2
(416) 569-1212
Price: $39.95
Inquiry #231

F40 Pursuit Simulator
Titus Software Corporation
20432 Corisco Street
Chatsworth, CA 91311
(818) 709-3692
Price: $44.95
Inquiry# 232

JGoodies_1
Delta Research
P.O. Box 1051
San Raphael, CA 94915
(415) 461-1442
Freely redistributable JForth software
Inquiry #233

Jinks
Hard Wired Entertainment
2171 Dunwin Drive, Unit 13
Mississauga, Ontario, Canada, L5L 1X2
(416) 569-1212
Price: $29.95
Inquiry #234

King James Version Bible on Disk
Easy Script
10006 Covington Dr.
Huntsville, AL 35803
(205) 881-6297
Price: $25.00
Inquiry #235

Professional Page Templates
and Design Guides
Gold Disk
P.O. Box 789, Streetsville
Mississauga, Ontario, Canada L5M 2C2
(416) 828-0913
Price: $59.95
Inquiry #236

Prospector in the Mazes of Xor
Eurosoft International
70 Woodfin Place, Suite 400
Ashville, NC 28801
(704) 255-7590
Price: $39.95
Inquiry #237

ProWrite 2.5
New Horizons Software
206 Wild Basin Road, Suite 109
Austin, TX 78746
(512) 328-6650
Price $124.95
Inquiry #238

SID: A Directory Utility
Software Ingenuity
11325 94th Street North
P.O. Box 10084
Largo, FL 34643
(813) 393-8240
Available as shareware;
$25.00 contribution required to become
registered user.
Inquiry #239

Structured ClipArt
Gold Disk
P.O. Box 789, Streetsville
Mississauga, Ontario
Canada L5M 2C2
(416) 828-0913
Price: $59.95
Inquiry #240

Super_DJ V2.0 Printer Driver
Creative Focus
P.O. Box 580
Chenango Bridge, NY 13745-0580
(607) 648-4082
Price: $25.00
Inquiry #241

Turbo
MicroIllusions
17408 Chatsworth Street
Granada Hills, CA 91344
(818) 360-3715
Price: $24.95
Inquiry #242

X-CAD Designer
Hazlitt Mews
Off Hazlitt Rd.
London W14 0JZ England
(01) 603-3313
North American distribution through:
American Software Distributors
(800) 225-7941
Price: $149.95
Inquiry #243

ProText's word processing features include an easy-to-use spell checker. You can spell-check an entire document, a block of text, or use it interactively as text is entered. There is also word search, as well as a user-definable dictionary. Text pages are automatically formatted as they are entered and edited. Foonotes can be entered into your document; headers, footers, and automatic page numbering is also available.

ProText also features mail-merging capabilities, allowing you to personalize documents based on information contained in a data file. Selective, conditional, and alternative text merges are all possible. Text editing features allow you to cut and past blocks of text from one file to another.

With ProText's command-line interpreter, you can perform desktop functions without exiting the program. You can run programs such as compilers, assemblers, and linkers from within ProText. The MichTron package offers a hassle-free environment, with all commands being accessible through menus, and full on-line help available at any time.

The program supports most dot-matrix printers, including 24-pin printers; 21-pin drivers are included. Several laser printers are also supported. A print buffer is also included, allowing you to use the program while printing a document. Odd and even pages can be printed separately to allow for double-sided printing.

MichTron's ProText runs on a minimum 512K RAM. The package includes the program disks and a hefty wire-bound manual.

ProText
MichTron
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700
Price: $199.95
Inquiry #226

•AC•

The Amazing Computing Freely Redistributable Software Library announces the addition of...

New Orleans Commodore Klub's

inNOCKulation Disk

Version 1.5

To help inform Amiga users of the newer Amiga viruses and provide them with the means to detect and eradicate those pesky little critters!

Files and directories on the inNOCKulation Disk include:

Virus_Texts (dir)
Various text files from various places (Amicus #24, PeopleLink, and elsewhere!) describing the Virus(es) and people's experiences and their recommendations; TVSB "The Virus Strikes Back": satirical text describing future efforts to rid the universe of the dreaded (silicon) viruses! Interview with the alleged SCA virus author!

WB_VirusCheckers (dir)

VirusX3.2
Runs in the background and checks disks for viruses or non-standard boot blocks whenever they are inserted. (Recognizes several viruses and non-standard boot blocks. Removes virus in memory. Has a built-in "view boot blocks" & other features.)

Sentry
Revision of VirusX1.01 in Lattice C.

ViewBoot
Highly active mouse-driven disk and memory virus-checker which allows you to look at the pertinent areas (useful in case you suspect a NEW virus!)

VRTest3.2
Watches memory for viruses; will alert the user and allow their removal if found. Can check & INSTALL disks, etc.

CLI_VirusCheckers (dir)

AVirusII
From The Software Brewery (W. German). Disables a virus in memory.

Clk_Doctor3
Corrects problems with the clock (caused by malignant programs, perhaps not really a "virus") (A500 & A2000)

Guardian1.1
Checks for attempts at viral infection at boot! Allows you to continue with a normal boot (if desired). Includes a small utility program to permanently place the program on a copy of your Kickstart disk.

KillVirus
Removes (any?) virus from memory.

VirusKiller
A graphically appealing and user friendly program by TRISTAR.

Boot-Block_Stuff

SafeBoot2.2
SafeBoot will allow the user to save custom boot sectors of all your commercial disks and save them for such an emergency. If a virus somehow manages to trash the boot sectors of a commercial disk, just run SafeBoot and it will restore the boot sectors, therefore saving your disk!!

Virus_Alert V2.0.1
Yet another anti-virus program with a twist. Once installed on your boot disk a message is displayed just after a warm or cold boot notifying the the user that the disk and memory are virus-free, and forcing a mouse-button press before continuing.

BootBack1
Saves and restores boot-blocks. Runs from CLI only.

Antivirus *aka AVBB*
Includes SEKA assembler source.

XBoot
Converts a boot-block into an executable file, so you may use your favorite debugger (Wack, Dis, ...) to study it.

The inNOCKulation disk also includes icons and arc files.

To order the inNOCKulation disk, send: **$6.00** *includes postage & handling*
($7.00 for non-subscribers)

Amazing Computing
inNOCKulation disk orders
P.O. Box 869
Fall River, MA 02722

HiSoft Compiler

by Cole Calistra

"Yet another BASIC compiler?", you ask. Well, yes and no. Up to now, we have seen old unreliable (AmigaBASIC), as well as A/C BASIC and True BASIC. HiSoft BASIC Professional, published by MichTron, Inc., is the newest package to hit the market. At first, it seems to be a combination of A/C BASIC from Absoft and True BASIC from True BASIC, Inc. However, HiSoft BASIC has more powerful features than either of these.

At one time or another, many of us have used AmigaBASIC, found on the Extras disk, and quickly realized that it could not be used for any serious application. The editor lacked many features, crashed often and, worst of all, was painfully slow. To a great extent, HiSoft has corrected these problems.

HiSoft pluses

This package includes a faster editor with some very good features, including search and replace and compiling from within the editor. In another helpful feature, after you have tried to compile your program, HiSoft will jump to the line that contains an error .

The editor is very quick and performs all operations with ease. There are no more delays when scrolling through your source code, and your program is not left hanging after an error. You can compile and run your program from within the editor, although this may require some added memory.

The compiler's combined linker is also fast and efficient. It links the modules itself for stand-alone code, although it can produce BLink-compatible object code. To check the efficiency of the startup code, I linked in with the Lattice C startup module (c.o) instead of letting the compiler do it for me, and it produced a larger code size than what I got with the HiSoft compiler. As for speed, it is one of the fastest compilers I have seen.

> *This package includes a faster editor with some very good features. As for speed, it is one of the fastest compilers I have seen.*

The compiler gives you the option of producing stand-alone programs, or programs that will run like a stand-alone but need the HiSoft.Library in the LIBS: directory. Note that the true stand-alone requires at least one megabyte of memory and the HB.Compiler will not produce true stand-alone on the first disk. You will need to use the HBS.Compiler on the second disk.

The language itself is about 95% compatible with AmigaBASIC, with a few simple differences. For example, all DECLARE .. LIBRARY statements must be made after the library is opened. Also, the LIBRARY CLOSE statement will not work. It will compile without errors, but when the program is run, it will crash. An interesting note on HiSoft's completeness is that, when the programs created with it actually do crash, there is a different GURU message that lets you return back to where you were after you hit the left button, instead of resetting the machine (a welcome change for many programmers!). There are other small differences between the two languages, but these shouldn't affect your programming too much.

HiSoft also took a step forward in their language in that it is not just compatible with AmigaBASIC, but with some of A/C BASIC's enhancements as well. They have also introduced some new statements of their own, while adding features to the existing ones in AmigaBASIC. An example is the SELECT CASE statements, which allow you to build a structure of different statements to execute if a variable is equal to a certain set of values, very much like the switch() statement in the C language. Two very useful statements are INCR and DECR, which will increment or decrement a numeric variable by one, thus replacing the traditional X=X+1 with INCR X. The major difference here is readability and speed.

HiSoft has also added a statement called PCOPY, which dumps the contents of the screen to the printer, like the ScreenDump program on the Extras disk. Additional enhancements include borderless windows, windows that are not GIMMEZEROZERO (a flag in Intuition that prevents you from drawing on the borders), backdrop windows, and a flag to the COLOR statement allowing you to set the drawing mode.

The down side

Although HiSoft is graced with features, it does suffer from some major problems. The OPEN statement does not support non-AmigaDOS device names, so COM1:, SCRN:, and LPT1: do not work anymore. Instead, HiSoft uses SER: for the serial port, PRT: for the printer, and CON: or RAW: for the screen. However, a problem may arise when changing the serial port parameters. SER: goes only by the preferences settings, thus making HiSoft impossible to use when writing a terminal or BBS program.

Further, the manual lists a FILL statement which does not exist in the compiler. In the revision, HiSoft gives instructions to disregard the manual and use the more powerful PAINT statement. Unfortunately, I received the newest version too late. It had been damaged in shipping and did not work anyway. Some other bugs in the program are the MENU OFF and MOUSE OFF statements, which simply did not work at all.

For technical support, Michtron has a bulletin board, a customer support number, and conferences on many of the major commercial networks. I used their BBS a few times to ask questions, and the responses were always quick and accurate. They also take suggestions for future versions of the compiler, which is good policy for any software company.

Summary

If you plan on doing some casual programming and don't mind the bugs in the OPEN command, I would recommend HiSoft BASIC Professional. I have used it extensively in developing applications software, and I believe it to be a very powerful tool for any programmer, beginner to expert. I would also recommend it to any avid AmigaBASIC fan. Its features and the program quality it generates are well worth the price tag.

•AC•

Product Information

HiSoft BASIC Professional
The Old School
Greenfield, Bedford
England MK45 5DE
Contact in US: *Apex Resources*
129 Sherman St.
Cambridge, MA 02140
(617) 876-2505
(800) 343-7535
Price: $159.95
Inquiry #221

SNAPSHOT

Four Amiga Games

by R. Bradley Andrews

Grand Prix Circuit

First on the list this month are two recent releases from Accolade: Grand Prix Circuit and Fast Break.

Subtitled as the "Formula One Racing Simulation," Grand Prix Circuit puts you in control of some of the fastest cars in the world. Each of the three cars is rated in three areas—speed, handling and braking ability. The Ferrari, while a bit slower, is the easiest to handle and is a good beginner's car. The Williams is in the middle and requires a bit more skill, while the McLaren is the most responsive, but also the most difficult to control.

Eight famous race tracks are available. From the Circuit de Monaco to the Detroit Grand Prix Circuit, each has a different layout and will require different driving strategies to master.

While you can practice on any of the tracks to learn how to shave every second of time off your score, most time will be spent running the single race. The first stage is a qualifying round, where one car's time is compared to the times of the other cars to determine its starting position in the 10-car starting grid. The actual race can be anywhere from a short sprint of one lap, to an endurance marathon of a full 99 laps.

In longer races, it is not just a matter of driving around and around the track; as in a real race, pit stops begin to play an important role. While pit stops cost time, the resulting increase in performance and handling is more than enough to make up for any loss.

For additional variety, difficulty levels range from the Rookie level where things are relatively easy, to the Pro level where everyone is out for blood. Once enough confidence has been gained on each race course, a complete tour of eight races can be done sequentially.

The main display features the competing car dashing along the bottom third of the screen, while the view of the windshield takes up most of the remaining screen. Two rear view mirrors are mounted just above the dash, allowing the driver to keep an eye out for cars approaching from the rear. The location of the car on the course is shown in the upper left corner, and the time is shown in the upper right. The graphics are crisp, clear and well drawn. The sound also compliments gameplay.

Gameplay itself is the game's biggest shortfall. While a steering wheel would be the best device to use in a racing game, a joystick can work OK. But the game is far to sensitive to slight joystick movements. It is very easy to end up weaving from side to side when trying to get centered back on the road.

It is also extremely easy to run into the another car when coming off of the starting line. One collision can end the whole race before it even gets started. While this is very frustrating, it might be acceptable if there were a quick way to get back into the race. Alas, even the "quick" start option still requires a long delay for the title screen to come up and then for the game to reload.

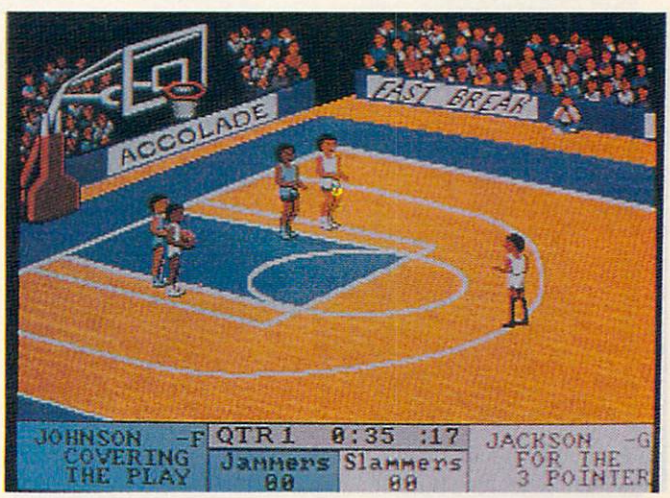
While Grand Prix Circuit looks like a very detailed simulation of racing, the control difficulties make it unsuitable for all but the most fervent racing fan to enjoy. If Accolade could fix the steering problem and the ease of collision at the starting stage, this might be a game worth looking at. But for now, I would recommend avoiding it.

Fast Break

The Slammers and the Jammers come head-to-head in a three-on-three variant of basketball where the focus is on action and well executed plays in Accolade's Fast Break.

Each game begins with the standard setup sequence. First, the time per quarter (3, 6, 9 or 12 minutes) is

Grand Prix Circuit: *Fast cars and fast reflexes on the track.*



Fast Break: *Slamming and jamming three-on-three.*

chosen. Shorter games play faster, but longer games add more challenge. Then you must decide which team the computer should control, or if it will "sit out" and let two players go head-to-head.

Two more decisions must be made before the game can begin. Decide which four out of the available 15 plays will be active at the start of the game. Though these can be changed during time outs and between quarters, take care to choose a set that will compliment each other and allow for the most scoring possibilities. One play slot can even be a play of your own design, allowing even greater flexibility. After the plays have been set, it is on to player selection. Each team has a total of six available players, two in each position of guard, forward and center. Matching the player's skills between themselves and against the chosen preset plays is important in order to be successful.

Defensively, five options are available during a game: Fast break, Man-2-Man (tight), Man-2-Man (loose), Trap, and Double Team. A wise coach will learn how to quickly switch to the most effective strategy for the action at hand.

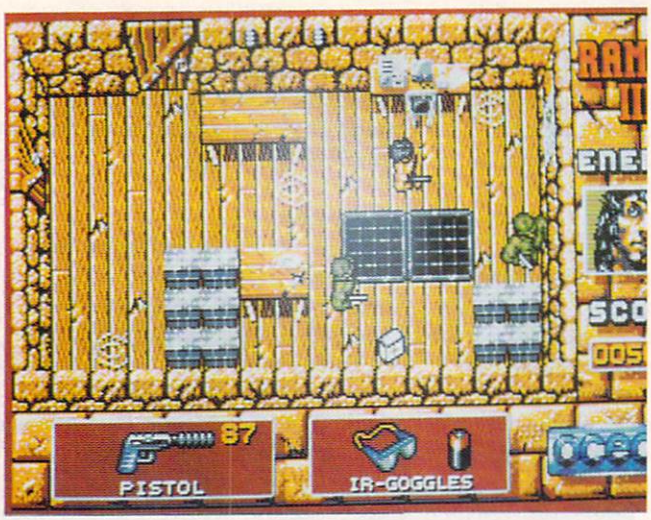
While the preset plays are available on offense, nothing prohibits a player from charging the basket if a lane opens up. While correctly implementing a fixed play will increase the chance of a successful shot, it is often necessary to change the call in mid-play. Shots can range from slam dunks to long distance three pointers. Rebounding also plays a role in both offense and defense and must be played aggressively.

Be careful not to be over aggressive. Both fouls and violations are portrayed in the game. Fouls can occur when either an offensive or defensive player has physical contact with another player. Violations are infractions of the rules, such as the 24-second shot clock, and the half court violation. Many normal problems, such as double dribbling, simply don't occur in the game since the computer handles the task of dribbling.

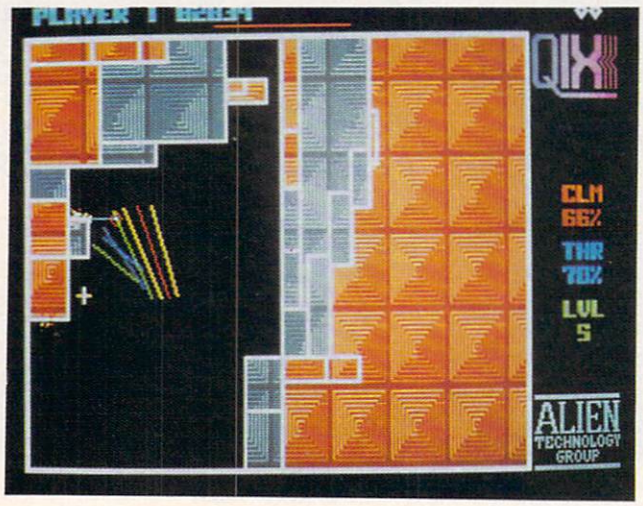
During the game, most of the screen is taken up with a view of the active half of the court. Along the bottom of the screen is a status area with such information as the currently active offensive and defensive players, the score, and the time remaining. The graphics themselves, while clear, somehow lack the zest found in many other action games. In fact, many of the starting screens where the game settings are set feature nothing but text on a colored background. The sounds are accurate, right down to the squeaking of hi-tops on the court floor.

As the rulebook states, you can use as much basketball strategy as you are willing to learn in Fast Break. However, I found this to be its biggest failing. While the game may be great for someone who dreams about the perfectly executed pick and roll, those of us who know little about basketball and do not have the time or inclination to learn a great deal are left in the dust. I also found the controls difficult to comprehend. Even the practice area was hard to master, and this is where I am supposed to learn how to play the game!

I am coming to the realization that I either really like, or really dislike an Accolade game, and unfortunately, I cannot recommend Fast Break. While the game has potential, it does not live up to the needs of most game players. If you are a knowledgable basketball fan, you may want to check out this game anyway; there is not much competition.



Rambo III: *The Soviet slime have captured your mentor, Colonel Trautment, and boy are you mad!*



Qix: *The arcade game makes it to the Amiga in true form.*

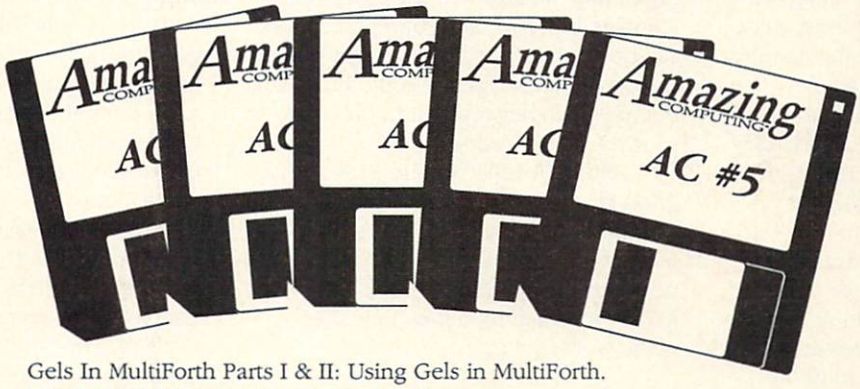
Rambo III

Next, it seems that the Soviets have captured your friend and mentor, Colonel Trautment. You are the only man alive with the ability to penetrate the enemy compound and rescue the man who taught you everything you know. So begins Rambo III, from Taito. Based on the movie of the same name, the game combines elements of arcade action with strategic planning as the daring rescue mission is carried out.

Beginning in the fortress, you must find where the Colonel is being held and release him. While this may sound simple enough, large numbers of guards patrol nearly every inch of the compound and you will need to carefully

A Winning Hand!
Amazing on Disk

Source Listings and Executables from the pages of Amazing Computing!
Only $6.00 per disk ($7.00 for Non-Subscribers)



AC #1
V3.8 & V3.9

Gels In MultiForth Parts I & II: Using Gels in MultiForth.
FFP & IEEE: Math routines in Modula-2.
CAI: Computer Aided Instruction in AmigaBASIC.
Tumblin' Tots: Save the falling babies in this game. Written in assembler.
Extra Goodies: Three freely redistributable programs- VGad, MenuEd & Bspread.

AC #2
V4.4

Fractals Part I: An introduction to the basics of fractals with examples in AmigaBASIC, True BASIC, and C.
Shared Libraries: Using shared libraries in C.
MultiSort: Sorting and intertask communication in Modula-2.
Double Playfield: Using dual playfields in AmigaBASIC.
'881 Math Part I: Programming the 68881 math coprocessor chip.
Args: Passing arguments to AmigaBASIC.

AC #3
V4.5 & V4.6

Digitized Sound: Playing digitized sounds using Modula-2.
'881 Math Part II: Part II of programming the 68881 math coprocessor chip using a fractal sample.
At Your Request: Using the system-supplied requestors from AmigaBASIC.
Insta Sound: Tapping the Amiga's sound from AmigaBASIC.
MIDI Out: A MIDI program that you can expand upon. Written in C.
Diskless Compiler: Setting up a compiler environment that doesn't need floppies.

AC #4
V4.7 & 4.8

Fractals Part II: Part II on fractals and graphics on the Amiga in AmigaBASIC and True BASIC.
Analog Joysticks: Using analog joysticks on the Amiga in C.
C Notes: A small program to search a file for a specific string in C.
Better String Gadgets: How to tap the power of string gadgets in C.
On Your Alert: Using the system's alerts from AmigaBASIC.
Batch Files: Executing batch files from AmigaBASIC.
C Notes: The beginning of a utility program in C.

AC #5
V4.9

Memory Squares: Test your memory with this AmigaBASIC game.
High Octane Colors: Use dithering in AmigaBASIC to get the appearance of many more colors.
Cell Animation: Using cell animation in Modula-2.
Improving Graphics: Improve the way your program looks no matter what screen it opens on. In C.
Gels in Multi-Forth-Part 3: The third and final part on using Gels in Forth.
C Notes 4.9: Look at a simple utility program in C.
Russell's Stuff: A collection of PD programs including: 1D_Cells, Colourscope, ShowILBM, Labyrinth_II, Most, and Terminator.

pick your way through the maze and make it to the next section.

After the Colonel has been rescued, it is off to the vehicle compound. Here, the enemy's vehicles must be neutralized by carefully placing demolition charges, escaping afterwards in a waiting helicopter. Finally, you face a run to the border. The chopper takes you to a waiting enemy battle tank which you must guide in a desperate race to escape the country before being caught by the enemy.

The guards are the principle hinderance to the successful completion of your mission. While most are fairly dumb and follow a fixed pattern of movement, walking into their view causes them to attack with extreme fury. And this additional noise draws other guards, which can further ruin your day. Fortunately, if enough attackers are slain, peace returns and continuation of the mission is possible. Electric doors and mines also block your path and can prematurely end the mission.

However, not everything in the game is harmful. While initially armed with only a knife and your wits, various useful items are scattered along your path. Stashes of both normal and explosive arrows can increase your ability to slice through foes. Medical Kits can restore lost strength. Other items, such as infrared goggles and rubber gloves can prove very useful.

The graphics in the game are done very well and rival those seen in traditional coin-op games. Some of the effects, such as the picture of Rambo turning white as strength is lost, are very detailed, and when combined with the smooth character movement, serve to draw the player into the playing experience. The sound compliments gameplay, though the background music can become tedious if left playing for a long time. The joystick is the principle input device and works very well.

However, the game is not all roses. A save game feature would have been really nice, since it can often take quite some time to complete even the first part of the game. The instructions are also very simple and lack some of the information needed for gameplay. Because of this, it is very easy to do something that will end your life without even realizing it, such as walking into an electric door. I even walked into a minefield and was destroyed even though I was carrying a mine detector.

Since each game only gives you one shot at success, death is an extreme bummer. I find little reason that such a long game should require restarting at the beginning after every death. Two to five lives would have been much more reasonable.

Much of the playing time involves imprinting the layout of the entire complex in your mind, so required tasks can be accomplished quickly. Drawing a map on a piece of paper could be useful, but most people would not bother to get so involved in a game.

On the whole, Rambo III is a very good arcade action game. While the quickness and finality of death can be most annoying, enough variety and enjoyment is provided to keep any gamer occupied for a long time. Definitely worth the money.

QIX

Finally this month is a new version of an arcade classic. Taito has finally brought QIX (pronounced kicks) to Amiga screens everywhere. QIX is different than nearly every other computer game and is a challenge to describe. (It is the game that inspired PowerStyx, reviewed last issue.)

All play in QIX occurs in a large rectangular area, a certain percentage of which must be boxed off to advance to the next level. Movement is restricted to the interior boundary of the enclosed area. This starts as the playfield edges, but these edges grow as more and more of the screen is filled in. Filling an area involves holding the fire button and pushing the joystick perpendicular to your current direction within the game boundary. The enclosed area can be any shape as long as it ends back on another interior edge.

Areas can be "drawn" at either slow or fast speeds. Slower speeds gain more points, but also place your token at increased risk. While drawing a box, you are vulnerable to contact with the Dix, the multi-colored line thingie that semi-randomly goes around the screen. If it comes into contact with either you or an unfinished box, you will lose a life. Once an area is enclosed, however, it becomes safe and may be used as a launching point for further boxes.

Two other enemies also seek to hamper your progress. SPARX also trace their way around the interior edge and any contact with them causes instant death. While they are relatively easy to avoid by drawing a new box, more and more are added the longer it takes to fill the required percentage of the screen, until a collision somewhere is almost assured. The other enemy is the SPRITZ. They are shaped like the sparks, but roam the interior of the playfield, and are only deadly while in the middle of drawing a line.

You might think it would often be good to start a line, then pause and wait for danger to pass. Well don't pause too long, since an unfinished border becomes a fuse if you pause for more than a short time. If caught by a fuse, you will die!

For such a simple game, it is still surprisingly fun. The graphics are relatively simple by modern standards, though the fill patterns are a little more detailed. The joystick works fine as a control and the sounds used flow with gameplay.

While not the world's best game, QIX would make a valuable addition to any arcade collection and is a fairly inexpensive way to spend some playing time.

•AC•

Grand Prix Circuit
Accolade
550 South Winchester Blvd.
San Jose, CA 95128
(408) 985-1700
Price - $49.95
Inquiry #201

Fast Break
Accolade
Price - $44.95
Inquiry #202

QIX
Taito
267 West Esplanade
North Vancouver, B.C.
Canada V7M 1A5
(604) 984-3344
Price - $39.95
Inquiry #203

Rambo III
Taito
Price - $34.95
Inquiry #204

PD Serendipity

Insight into the World of Freely Redisributable Software for the Amiga™

by Mike Morrison

Fred Fish Disk #229

AlarmingClock: An alarm clock that plays two digitized sounds when it goes off. One is an explosion and the other is a man screaming. I took the author's advice and played it through my stereo at half volume. It would definitely wake me up, and the neighbors, and the guy down the block. In the read me file, the author says that the sounds are IFF but he reads them as hunks. I tried a few of my own sounds that were straight data samples that still sounded close to the actual sound. I had to rename my sound to one of the two orignal sounds in order to get it to work. Includes source. Author: Brian Neal

DrawMap: This program generates pictures of the Earth as if you were looking at it from space. It can generate several different views: flat, mercator, globe and orbital. You can pick a spot on the Earth for it to use as the center of the map it generates. It takes an average of 35 seconds to draw each view. Includes source. Author: Bryan Brown

Emporos: A text adventure game. You are living on the island of Emporos, where several countries exist. Your goal is to make one of these countries your own. Binary only. Author: Roland Richter

esuoM: This program changes the direction of the mouse to opposite of the input. It can be a fun joke to play on someone. Includes source. Author: Rob Eisenhuth

LeftyMouse: This program swaps the left mouse button with the right, and vice-verse. A nice program for left-handed people. Includes source. Author: Rob Eisenhuth

Shuffle: A program that will shuffle all the screens that are open. When you use Left-Amiga-M, the front screen will go to the back. Includes source. Author: Rob Eisenhuth

Sim: This program was a little out of my ball park so here is an excerpt from the description that comes on the disk: A simulator for register-transfer nets, which are used to describe hardware systems. This version also provides a compiler to define new devices in addition to Sim's internal devices. Version 4.0, binary only. Author: Gotz Muller

Fred Fish Disk #230

AskTask: Shows all of the tasks in the system attached to ExecBase. You can then examine different bits of the task structure. Displays priority, state, flags, stack, signals, etc. You can remove tasks, change the priority of a task, or send arbitrary signals to a task. Removing tasks can cause the system to come tumbling down, so be careful. Version 2/4/89, includes source. Author: J. Bickers

Fedup: See Expansion Drawer.

FileIt: A database program that was first prototyped on a CP/M system in Turbo Pascal 3. Eventually it was ported to the Amiga and translated to DRACO. The author suggests that an Intuition interface should be added in the next upgrade. Version 1.0, includes source. Author: John Davis

NComm: A telecommunications program based on Comm version 1.34, by DJ James. The program has been enhanced with many new features and rivals some of the commercially available telecommunications programs. Included are several auxiliary programs such as AddCall, CallInfo, IbmIso, PbConvert, and ReadMail. This is version 1.8, binary only. Author: DJ James, Daniel Bloch, Torkel Lodberg, et al.

PrivHndlr: A privilege violation handler for the 68010 CPU. This program is similar to Decigel but will survive a reboot so you can use it with copy-protected programs that run from boot. Version 3, includes source in assembly code. Author: John Veldthuis

Quattro: A Tetris clone. This program plays the same as the other Tetris-like games but has a few extras. It has three levels of play, sound effects, a 43-color background, next stone preview, and joystick or number pad control. Version 1.0, binary only, source available from author. Author: Karl-Erik Jenss

Fred Fish Disk #231

Diff: Another diff program. This one implements the algorithm from *Communications of the ACM*, April 78. Has enhanced output. Includes source. Author: Donald C. Lindsay

File: A program that looks at a file and tries to figure out what its type is. Recognizes font files, icon files, executable files, standard object files, compressed files, command scripts, C source, directories, IFF files, LaTeX source, Modula II source, arc files, shell commands and scripts, TeX source, dvi files, uuencoded files, yacc files, zoo archives, etc. Version 1.0, includes source. Author: Edwin Hoogerbeets

NoClick2: A program which stops the empty drives on the B2000 from clicking when using AmigaDOS 1.3. It should also work on an A500. Binary only, source available from author. Author: Norman Iscove

Plot: A program that works with

Expansion Drawer

Fedup Fred Fish Disk # 230

This program is a file editor. You can load either binary or ASCII files to look at or edit. It is best used for binary files. The program assumes a record length of 256 bytes. If the file does not end with an even 256-byte record, the program will pad the empty space with the # sign. Of course, you can't edit this area.

When you start Fedup, an open requester comes up. Once you have selected a file to load, you can then move through the file by records. Status information is displayed on the screen, such as current record, last record, and the name of the file you are currently working on. The program uses the arrow keys to move around. The file is displayed in both hex and ASCII. You can choose to edit in either hex or ASCII. A nice feature of Fedup is that when you use the backspace key instead of deleting a character as normal, the original value is replaced. This is good if you make changes and then decide it is a mistake. Simply place the curser over the last character you changed and hold down the backspace key. All the original characters will be replaced.

The program has menus that allow you to do things like write the record you have changed, undo the last edited record, go to a certain record, search for a string, print the current record, choose between using all 256 ASCII characters (allowing ALT characters) or just the lower 127, and the obvious info, open, and exit. There is also an autosave option that will save each altered record when you move to another. The default will not save the records you change unless you tell it to with the write menu selection.

One of the fun things you can do with a binary file editor like Fedup is customize your commands. Remember to do this with copies of your original commands in case you make a mistake. I loaded in the NEWCLI command from my C directory. Then I searched for the string "New CLI". This is the text that appears in the title bar of each CLI you open with the NEWCLI command. Then I changed the text to "Mike's" and saved the record. You have to be sure to only use the same number of characters that are in "New CLI". This is the alloted space in the file and you will cause all sorts of problems if you go beyond the original length. Now when I use NEWCLI, they have "Mike's" in the title bar instead of "New CLI". The coordinates for the location and size of the new CLI could also be changed using this editor.

This program is version 2.1, binary only, and was written by Martin Lindemann of Norway.

MultiPlot and ThreeDPlot to help make 2D and 3D plotting easier. Plot calls MulitPlot and ThreeDPlot while you are working in Plot. AG Baxter wrote this interface and Tim Mooney wrote MultiPlot and ThreeDPlot. This is version 1.2 and includes source to Plot. Author: AG Baxter, Tim Mooney

Sed: Sed (stream editor) is the GNU sed program, ported to the Amiga. This program will automatically perform certain editing operations on the file you pass it or the standard input. You can specify the editing operations by the command line or a script file. Version 1.02, includes source. Author: Unknown, ported to Amiga by Edwin Hoogerbeets

Fred Fish Disk #232

BallyIII: This is an Amiga port of the arcade game named Click (similar to Qix). This version fixes some bugs found in Bally II (found on Fred Fish Disk #221. See "Expansion Drawer" AC V4.8, page 20.) Binary only, shareware. Author: Oliver Wagner

Dbug: This is a debugging package put together by Fred Fish. I don't know where he gets the time! Best said in his own words, Dbug is: Machine-independent macro-based C debugging package. Provides function trace, selective printing of internal state information, and more. This is an update to the version released on disk 102, and now includes a machine independent stack use accounting mechanism. Includes source. Author: Fred Fish; profiling support by Binayak Banerjee

ReSourceDemo: A demo of ReSource, an interactive disassembler for the Amiga. In this version the "save" feature has been disabled. This is version 3.06, an update to version 0.36 from Fred Fish Disk #192. Binary only. Author: Glen McDiarmid

Fred Fish Disk #233

Brik: This program calculates checksums for a file. It will do both text and binary CRCs (cyclic redundancy codes). Text

mode CRCs calculated by Brik are portable across systems for files that are in the usual text format on each system. Binary mode CRCs are portable for files that are moved from system to system without any change. Brik can verify and automatically update an embedded checksum header within the a file. It runs under MS-DOS, UNIX system V, BSD UNIX, VAX/VMS, and AmigaDOS. This is version 2.0 and includes source. Author: Rahul Dhesi

CacheCard: Another program from the labs of Dave Haynie that only he could write. CacheCard is in the words of the docs: An accessory to SetCPU for use with A2620 cards or 68030 systems. It modifies the MMU table set up by SetCPU to selectively control caching for each expansion card. It's also an example of how an accessory program can track down and modify the SetCPU MMU table without having to read all kinds of MMU registers and figure it out for yourself. Version 1.00, includes source. Author: Dave Haynie

CrcLists: Complete CRC check files for Fred Fish Disks 001-231 using the Brik program (also on this disk). These were made directly from Fred Fish's master disks. Fred has switched to Brik, from the CRC program used to make the lists on disks 133, 146, and 173, because, "It has more features and because source is available." Author: Fred Fish

Fred Fish Disk #234

Siebert Mania: This should be called the Fridtjof Siebert disk! All of these programs where written by Fridtjof. The best part about these programs is that Fridtjof also includes the source so we can have fun and learn too! Thanks a lot Fridtjof, keep up the good work!

KwikBackUp: A nice hard disk backup program that writes data track by track onto multiple floppy disks. It uses the archive bit, saves and restores comments and protection flags, and skips over bad spots during restore instead of aborting. Version 1.0, includes source in Modula-2. Author: Fridtjof Siebert

MuchMore: Another program that displays ASCII text files to the screen (like more, less etc.). This program opens a screen the size of your Workbench so it works with both PAL, NTSC and overscan. Allows up to 4 colors, bold, underlined, and italic text. Has a nice smooth scroll (forward or backward), search, and on-line help. Version 1.8, includes source in Modula-2 and Assembly code. Author: Fridtjof Siebert

NetWork: A neat screen hack. Be patient and let it run. Version 1.0, includes source in Modula-2. Author: Fridtjof Siebert

PrintIt: A program to print IFFs to an Epson-compatible, 9-pin printer. Prints in several resolutions (60-240 horizontal, 72, 144, or 216 vertical). You can convert color pictures to black and white by using several different options. Version 1.0, includes source in Modula-2. Author: Fridtjof Siebert

WBPic: Replaces Workbench's color 0 with a 2 or 4-color IFF picture that is the same resolution as your Workbench. Has some neat effects, but like Fridtjof says, he's not sure of a use for it! Version 1.0, includes source in Modula-2. Author: Fridtjof Siebert

XHair: Replaces the mouse pointer with a crosshair that goes off the top and bottom of the screen. This is nice for lining things up. There are also two other programs that change your pointer included. Version 1.0, includes source in Modula-2. Author: Fridtjof Siebert

Fred Fish Disk #235

CalcKey: A memory-resident calculator that pops up when you hit a hot key sequence. It supports decimal, octal, and hex which can be handy for converting numbers while in another program. A nice feature is that it will output the answer into the program you are running when done. If you have your word processor running and you want to do some math, you pop up the calculator and, when your done, it will put the answer where you cursor is in the letter you're typing! Version 1.0, binary only, shareware. Author: Craig Fisher

Ct: An Amiga program to display images from a CT scanner, along with several new interesting sample images of scans of real people. The display software, though it has a primitive user interface, is quite powerful, including functions like convolutions, averaging, laplacians, unsharp masking, edge detection, gradients, etc. This is version 2.2, an update to the version on disk 137. Binary only. Additional image disks available from author. Author: Jonathan Harman

MirrorWars: A new game featuring sound, title music, and two-player mode. You fight your opponent via laser rays, but beware of the mirrors reflecting your shots. Binary only. Author: Oliver Wagner

Fred Fish Disk #236

AmigaBench: Optimized Amiga assembly versions of the Dhrystone benchmark. Includes 68000 and 68020 versions. Author: Al Aburto

DiskHandler: A sample implementation of a file system that reads and writes 1.2 format diskettes. Includes source. Author: Software Distillery

Heart3D: A program to find left ventricle outlines in the output of an Imatron CT scanner, and display wireframe animations of the beating heart. Includes several sample CT scan outputs. Binary only. Author: Jonathan Harman

Ls: Version 3.1 of the popular UNIX style directory lister. This is an update to version 2.0 from disk 178, and includes some bug fixes, support for multiple wildcard pathnames, quicker sorting, a best-fit output, new output width and height options, and some other new features. Includes source. Author: Justin V. McCormick.

Proc: Example program of how to create a full-fledged DOS process without needing to call LoadSeg first. Based on an idea presented at BADGE. Includes source. Author: Leo Schwab

XprZmodem: An Amiga shared library which provides ZModem file transfer capability to any XPR-compatible communications program. Version 1.0, includes source. Author: Rick Huebne

•AC•

No Fishing!

The Ones That Got Away. ***A Closer Look at PD Software Not Currently in the Fred Fish Collection.***

by Graham Kinsey

Welcome! After the distress of seeing Commodore Magazine fold its tents, its nice for this column to have a home again! For everyone already familiar with this column via past installments in Commodore Magazine, let me say that not much will change due to the transfer to Amazing Computing, except that I will not be reviewing Fred Fish disks here (since they are covered in the PD Serendipity column). Instead I will be concentrating solely on programs obtained off of PeopleLink and local BBS's.

This month's features include Jack Radigan's entry into the terminal program world, the latest version of ARP, and more. For each program, the author will be given — when known, and in most cases the AmigaZone download file number will also be listed (this doesn't mean that if no file number is given that it isn't on PeopleLink at the moment, simply that I obtained it from somewhere else), so those individuals having access to PeopleLink and its AmigaZone can quickly download the file.

When a public domain program has been classified as shareware, this is also mentioned, with the suggested amount if given. Due to the large size of animations that are coming out for the Amiga, I have decided to assume that the normal size of an Amiga animation is one megabyte. This means that unless I specify otherwise, ALL animations reviewed here require one megabyte in order to run. For Amiga owners with only 512K machines, keep this in mind until more memory can be added to your Amiga.

ARP version 1.3

ARP version 1.3: by Charlie Heath (and many others) (AmigaZone files 16909 and 18142). In addition to being updated to provide maximum compatibility with version 1.3 of AmigaDOS, new features have been added. By far the biggest addition to ARP is the fantastic ARPInstall program. This gadget-laden program (written in Modula-2) not only makes it possible for Workbench-only users to install ARP, but makes it VERY easy to do, as well as making the process almost totally idiot-proof! Since the ARP commands are stored internally within the ARPInstall program itself, all a user needs to install and use ARP is the ArpInstall program. Another new program in ARP 1.3 is ASH. It is a shell with all the features that the Commodore's Shell supports, plus other features like built-in batch commands and command substitutions.

As for the ARP commands themselves, Copy will not abort when a read/write error occurs, it informs the user of the event and asks to continue. Copy also has a new mode that will not copy a file if an identical file exists in the destination path (some control over what determines which files are identical is given to the user). Assign and Mount can now handle multiple assignments/mounts on one line, which speeds up startup-sequence execution greatly. The type command can now add formfeeds to any file that is printed. ARP's version of Avail includes an option to flush all libraries and other resources from the system. Now there are several new ARP commands, including Move (which combines Copy and Delete), Cmp (binary comparison of files), Set (for changing environment variables), and LoadLib (Loads a specific library).

Listed in the documentation for ARP 1.3 (which was released several weeks after the commands) are many small incompatibilities between ARP 1.3 and AmigaDOS 1.3. While this has caused some to badmouth ARP on the networks, don't be discouraged from using ARP! Most of these discrepancies are things average users will never care about, and they are no reason for not taking advantage of the reduced code size and powerful features provided by ARP (Hard drive owners might want to keep both the AmigaDOS and ARP commands on their hard drive, in separate directories, for convenience).

JRComm

JRComm: by Jack Radigan (AmigaZone file 18208; Shareware: $30). Jack Radigan, the author of the famous Online/PCPursuit scripts, has entered the highly competitive terminal program world with his entry, JRComm. JRComm could best be described as a streamlined and efficient terminal program, which does not have all the fancy features (and colors) that Access! has, but instead it is designed for those who have little time to waste while online. JRComm has a long list of available protocols, including the only full-featured terminal program to include Zmodem internally (since many wouldn't consider AZComm to be a full-featured program), and the ONLY terminal program to support Ymodem-G (which is very important for those with

MNP-compatible modems). As might be expected, JRComm is no slouch when it comes to transfer throughput; only AZComm gives faster ZModem transfers than JRComm.

However, JRComm is superior to AZComm, making it much easier to perform batch transfers. Not only does JRComm's superb batch file requester allow for the selection of any number of files (even across devices!), but it also lists all files that have been selected. JRComm also allows control over which files will be uploaded during a Zmodem transfer. JRComm can decide whether to overwrite (or append) an existing file on the receiver's reception directory according to file size, date and/or CRC value. Some of JRComm's other features include complete ANSI color support, as well as support for overscan and PAL environments.

JRComm also has a unique phone directory supporting parameters for phone entries that other terminal programs don't. In addition to fields for capture buffer and macro file names, there are also fields for a cents per minute (for online services) and password fields. The password field is very useful since there is a function in JRComm that can send the password out the serial port to the awaiting BBS or online service. The biggest problem with JRComm also relates to it's phone directory. It does not support existing phone directories store in The Final List format. Although Jack Radigan has promised to add a conversion program to JRComm for TFL-format phone directories, he has said that it will only be available for registered users. Other than this one problem, JRComm is a very good terminal program that is at least in the same league with the current king of the hill Access!, if not better than it. In particular those who are using AZComm now should take a close look at it, especially if transfer throughput isn't the only thing you care about.

As of version .94a of JRComm, Jack Radigan has stated that this version would be the last freely distributable version for quite a while, so be prepared to pay the shareware fee if you want to see any future versions of JRComm until the end of 1989.

Dither8

Dither8: by Eric Quackenbush (AmigaZone file #16707). A simple graphics demo that demonstrates how to display over 1000 apparent colors in 640x400 mode using a combination of dithering and palette-changing techniques. Although source code isn't provided, this program is useful in that it demonstrates how to display more colors than normally possible with the Amiga. Now that the Amiga has been out for four years now and programmers are now starting to push the limits of the machine's hardware capabilities, tricks like this are vital to the improvement of graphics-intensive software for the Amiga.

InTouch

Intouch: by Khaled Mardam-Bey (AmigaZone file #16448). Yet another derivative of Communicator v1.34, whose main additions include a scroll buffer, ANSI and VT100 support (copied from version 2.8 of Wecker's VT100 program). InTouch's other added features include variable chat window size (for the split screen option) and the ability to clear the screen, and a no-frills auto-redial mode. Needless to say InTouch is no threat to the big boys, yet having another choice never hurts.

In addition to the tiny executable size, QMouse supports residency, which may be enough for some people to adopt it.

Invention

Invention: by Durwood Trasher (AmigaZone file #16394). A very simple yet amusing animated story created with The Director. Invention is all about an inventor who's new idea isn't exactly greeted with excitement by the family pet!

Jumble

Jumble: by M.D. Groshart (AmigaZone file #16419). Those who like to solve anagram puzzles may like this program. It uses brute force to help solve puzzles by listing all options, but there is one filter option that can help to reduce the possibilities.

Jeopard

Jeopard: by Robert Caspar (AmigaZone file #16625). Jeopard is a very good Amiga version of the popular board game Risk. Jeopard lets two to eight (Sorry, no one-player option is available) people battle over the familiar colored land masses that make up the Risk game. All rules are faithfully observed, plus there are a few options that might not be expected.

The biggest addition is the AutoRoll option that allows an attacker to tell the computer to roll the dice for both sides automatically until the attacker's forces have dwindled down to a certain number (or until the defender is wiped out from the territory). There are also a bunch of options to alter or set-up the game board, including changing ownership of a territory, or changing the number of armies in a territory. Of course games can be saved when you can't devote hours at a time to Jeopard. The board is rendered in 640 by 400 mode, which gives it a clean look. The only problem with doing that is the program is written in compiled AmigaBasic, it takes a few minutes to set-up the game on a 68000. Despite that, any Risk fan will want to get ahold of Jeopard immediately.

MyMenu

MyMenu: by Darin Johnson (AmigaZone file #16552). Another program that allows for the addition of extra menus to the Workbench menus. The menu set is stored as a text file in the S: directory. It provides for the additon of sub-menus and key board shortcuts for each menu option. The pen colors can be changed at any time, and arguments are supported for CLI-based programs.

PopCLI_IV

PopCLI_IV: by John Toebes (AmigaZone file #16420). Version four of the first ever multi-utility utility program, PopCLI. Newest added features to PopCLI include ARexx support, multiple hotkeys (each with different command options. PopCLI will also grab several pieces of information from the system upon loading, including correct directory, stack size, the command line prompt and the fail level.

QView

QView: by Lyman Epp (AmigaZone file #16477; Shareware: $10). From the

author of QMouse comes this text file reader. In addition to the tiny executable size, QMouse supports residency, which may be enough for some people to adopt it (after all there still isn't an established replacement for Blitz, which first appeared over three years ago). Unfortunately QView has no file requestor nor menu set whatsoever, putting QView more on par with the likes of More and Less than Blitz.

TNT

TNT: by John Schieb (AmigaZone file #16612). Short for Telecom Network Timer, this program will keep a log of how much time a user spends on a telecommunication service each month. Since TNT does not accept online charge rates as a parameter it can't be used to calculate a bill, simply to calculate how many hours were spent on the service. Since TNT can keep track of time for two services at once, it can also keep track of how much time is spent on a host packet-switching network (i.e. Telenet/PCPursuit) while accessing a service that can be accessed from a host network (like PeopleLink).

WipeDemo

WipeDemo: by Paul Falstad (AmigaZone file #16432). A simple slideshow program that will display IFF pictures using 35 different wipes. Although the program has no other major features (except for script support and a few basic commands to provide a decent amount of flexibility in putting together a nice slideshow), the source code is provided for those who would like to see how wipe effects are created.

IconMeister

IconMeister: by Michael Bodin (AmigaZone file #16646; Shareware: $15). Another icon editing program. IconMeister uses an interface vaguely similar to DeluxePaint (except for the tool bar lying on the bottom of the screen), and its features include eight color support, animated icons, cut and paste and undo support.

Radio

Radio: by Larry Crandall (AmigaZone file #16131). An excellent Sculpt 4D animation of a radiometer (the four-paneled solar-energized device that "appears" to move for no reason). Another example of how the price tag on Sculpt 4D can be justified. This animation will run in only 512K of memory.

MiddleButton

MiddleButton: by Micheal Sinz (AmigaZone file #16976). This program allows for the use of the third button of a Boing mouse or other alternative "mouses" for the Amiga (without having to buy X-Windows). MiddleButton will make the middle button act as an extended select in the Workbench environment (i.e. no more holding down the shift key).

Checkers

Checkers: by Ronnie Pertuit (AmigaZone file #16808). A checkers game for the Amiga has finally arrived. This program, written in compiled AmigaBASIC is a no-frills version, with four difficulty levels as its only significant feature. Nevertheless, another classic board game can now be played on the Amiga.

Next month's reviews will include a new animation from Dr. Gandalf, plus updates to PcPatch, Showiz and much more. I can be reached on the AmigaZone on PeopleLink (ID: G KINSEY), or on the IDCMP BBS (617-769-3172 (3/12/2400 baud, 105 Megabytes online, running 24 hours a day), addressed to SYSOP). If you have written a public domain/shareware/freely distributable program, or have obtained one that you think is worth mentioning to all Amiga owners, then please attempt to contact me via the above contacts, or through Amazing Computing. See you next month.

To signup to PeopleLink and their AmigaZone, call them at: 1-800-524-0100 (voice) 1-800-826-8855 (via modem) For information on obtaining some the programs that aren't listed as being on PeopleLink (or for those who don't have a modem), please write (and/or send $2 for an Amiga PD catalog disk) to: SMAUG 1015 So. Quincy Ave. #112 Quincy, MA 02169

•AC•

It's the year 2000. Your city currently has several problems which include—but are not limited to—a rising crime rate, heavy pollution, an inadequate mass transit system and nuclear power plants that are much too close to major residential areas. The citizens are demanding a new seaport, as well as a new football stadium. To make matters worse, the administration is facing a budget deficit, and the mayor's popularity is dropping at the polls. Of course, there is one other minor complication. You just happen to be mayor.

SimCity is a fun and interesting city simulation program that puts you in charge of planning and managing a city. You can either start from scratch, or take over ready-made simulations which, for instance, put you in charge of Boston, Massachusetts right before a major nuclear meltdown. Other scenarios include managing Rio de Janeiro during a major flood or Hamburg prior to an Allied bombing run.

After booting your computer with Kickstart, insert the SimCity Disk. If you are using a printer driver other than the Epson driver provided with SimCity, you must also insert your Workbench disk. Double clicking on the SimCity icon reveals three icons: the SimCity icon, Last Minute Notes, and a Zone Evolution icon which provides a guide to the value of properties as they grow. To start the game, double click on the SimCity icon. From the startup screen you can load a previously saved scenario, a preset scenario, or a new game.

A new game starts after the computer has "terraformed" a large parcel of land. You have the choice of accepting the terrain, or having the computer terraform another parcel of land. Once you are happy with the land, name your city and choose one of three difficulty levels: Easy, Medium, or Hard. The Easy level gives you $20,000 to develop the land, the medium level starts you with $10,000, and the hard level grants $5,000.

On the left side of the screen you will see a small portion of the total land available (in the edit window), while on the right you will find a series of 16 icons from which to choose your actions. Move around the available land by moving the mouse to the far right, far left, top, or bottom. These movements cause the land to scroll in the same direction as the mouse. At the top of the screen is the city's name, the date (which always starts in January of the year 1900), and the amount of money you have to spend. Below the title bar is a message bar through which the Sims (the SIMulated citizens of your town) make known to you their requests.

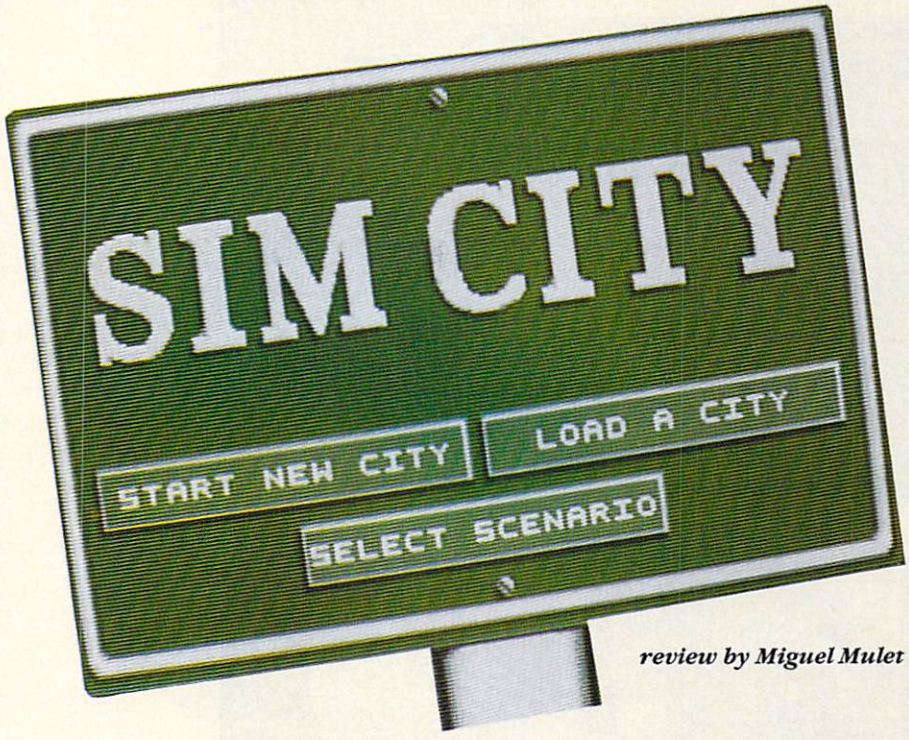
Using the icons on the right, you can select from a number of possible actions—for a price. Bulldozing allows you to rezone, but it will cost $1. The road icon lets you build a small section of road for $10. As you go down the list, the prices increase. You can build parks; zone commercial, industrial, or residential areas; build fire and police stations; or create stadiums, seaports, or airports. Once an icon is selected, a small empty box appears at the end of your cursor, representing the area affected by the command.

To zone a piece of land as residential, hit the residential icon, move the cursor to the area you want zoned residential, and press the left mouse button. The other icons work the same way. If you make a mistake, you can undo your last action by selecting UNDO from the edit menu at the top of the screen. (You access the menus by pressing the right mouse button.)

The goal of the game is to create a living, thriving community of Sims. To achieve this end, you must provide jobs (through industry), homes (by zoning residential areas), and places to conduct business (in commercial zones). The Sims take care of building churches, hospitals, office buildings, and factories, but you must provide them with roads and mass transit. Property values increase and decrease depending on variables like access, pollution, and crime rate. As the population increases, you will also need to zone land for police and fire stations, airports, seaports, and stadiums.

Easy, you say. Well, what makes life difficult is that roads, fire stations, and police stations cost money to maintain. Of course, this money comes out of your budget. You can raise funds by levying taxes between 0 and 20%. Then, on at least a yearly basis, you decide how much money you would like to appropriate for your transit, fire, and police departments. Full coverage for each fire and police department costs $100 per year. The transit department requests funds in direct proportion to the amount of roads, power lines, and transit tracks you have built. If you provide too little money to the transit department, your transit system deteriorates and traffic problems ensue.

To add to your woes, the Sims themselves often voice complaints.



review by Miguel Mulet

Through the message bar, they communicate to you their desire for stadiums, more police or fire protection, etc. They also evaluate your performance as mayor to let you know what they think of your policies.

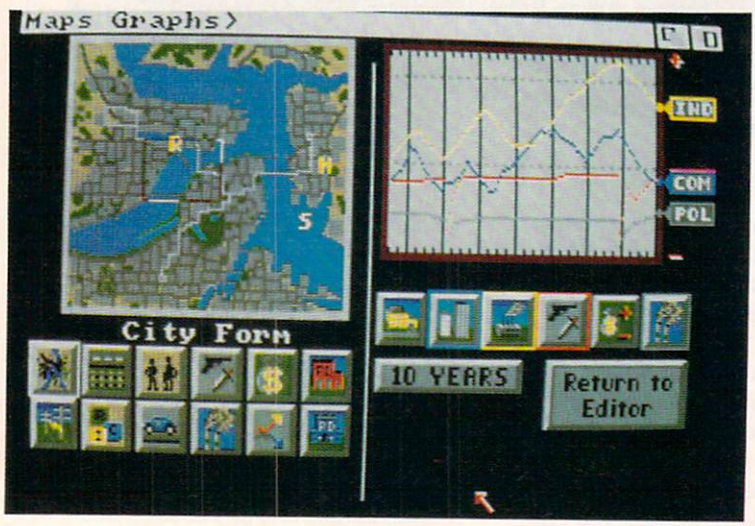
There is one other screen available to aid you on the job, and that is the maps/graphs window. You can access this screen by selecting either the map or graph icon on the right side of the editor screen. The map section of this screen is on the left, and the graph section on the right. The icons on the map section let you see the shape of your city, property values, zoning, population density, high crime areas, high pollution areas, growth areas, as well as police and fire protection. The graph area allows you to see general trends for your city, plotted for the short term (the last 10 years), or the long term (the last 120 years). You can plot the residential, commercial and industrial growth rates, as well as the crime, pollution, and population rates. Using these tools, you can see where your city is heading, and what actions you can take to help ensure its survival.

There are five menus that appear when you press the right mouse button. Using the FILE menu, you can start a new city, load, print or save a city, pick a preconfigured scenario, or quit. The EDIT menu allows you to UNDO your last command. The OPTIONS menu lets you set several variables. You can turn on the Auto Budget, which instructs the computer to use the same values for tax rate and expenditures every year. The Auto Bulldozer allows you to place zones on trees without bulldozing first. The Auto GOTO Event automatically takes you to the scene of major events. The SOUNDS command lets you turn the sound on and off, while the GAME SPEED allows you to pause the game, or set its speed to slow, medium, or fast. The POWER BOLTS command lets you turn off the flashing electric bolts which appear when a zone loses power.

If you get bored, look at the DISASTER menu. Using this menu, you can wreak havoc on your community by starting fires, floods, tornadoes, earthquakes, or air disasters. You can even precipitate a visit from Godzilla! The final menu is the WINDOWS menu, which allows you to change the Budget, seek an evaluation of your job performance and city statistics, or switch to the maps/graphs window.

The program has several hardware requirements. Your system must have at least 1 meg of RAM, Workbench 1.3 (if you want to use the print functions of the program), 1 disk drive, and a color monitor. The halfbrite chip is recommended, but not required (since my system has the halfbrite chip, I was unable to judge how the game would appear without it). Also helpful is a printer, so you can produce either a poster of your entire city (6 pages large), or a "snapshot" of a particular area. The program supports multitasking, memory permitting. Lastly, the game can be stored on a hard disk, although you will need to keep the original disk handy as the program uses a "KEY DISK" form of copy protection.

SimCity makes fairly good use of the Amiga's graphic and sound capabilities. Once you have zoned land, the Sims move in almost immediately and construct homes, schools, and other structures right before your eyes. As your city grows, more cars appear on the streets. Once things really get hopping, a railcar begins traveling along the rail lines. Should you forget to check what the latest crime statistics are, the digitized



Top: *The Bulldozing icon prepares land for new development.*

Bottom: *Maps and graphs make keeping track of property values, zoning areas, population density, high crime areas, high pollution areas and growth areas simple and fun.*

sounds of police sirens will alert you to any problems. Once you have built an airport, there is even a traffic helicopter to alert you of congested road areas. Also, once you have built a seaport, don't forget to look for ships along the waterways.

Alas, no program is perfect, and there are a few things in SimCity which could be improved. After playing the game for a while, I felt that even the "Fast" speed was too slow. I found myself spending a lot of time just sitting around waiting for time to pass.

Another problem is that, although the user manual says SimCity was written for the Amiga, the diagrams in the manual are screen shots from the Macintosh version of the game. For instance, the GOTO button is shown in the manual as an eye, although it appears on the Amiga screen as an arrow in the upper right corner. Several other diagrams in the manual do not correspond with objects on the Amiga screen, but these discrepancies do not significantly affect play.

Also, if you have the Auto GOTO Event turned on, be careful. You could be bulldozing one area and suddenly find yourself being automatically transported to the area of an event, where you may inadvertently continue to bulldoze. Even though you can fall back on the UNDO command, it will not always undo all the damage.

Overall, SimCity is an excellent game and city simulator. The designers have managed to keep things relatively simple and do not bog you down with a lot of unnecessary details. The game is entertaining, and if you get bored with your city, you can always build another one. As a simulator, SimCity shows you just how hard it is to plan and run a city. Well, duty calls. The demands of the citizens must be met!

•AC•

SimCity
Maxis Software, Inc.
953 Mt. View Dr., Suite 113
Lafayette, CA 94549
(415) 376-6434
Price, $44.95
Inquiry #209

Conference Simulation:

SimCity Creators Talk Railways, Zone Evolution, and Version 1.1 On CompuServe

edited by Richard Rae

On July 19th, 1989, CompuServe's AmigaForums hosted a formal conference on SimCity. The featured guests were Will Wright (WW), the city simulation's original designer, and Brian Conrad (BC) who, along with Brian Witt, developed the Amiga version. The following is a edited transcript of that conference. Names in parentheses represent conference participants.

AC: Welcome all to a special conference with Will Wright and Brian Conrad of SimCity fame. If you've been reading the forum's message base over the last month or so, you know that SimCity is the current "hot topic", being a "city simulator" which is a very addictive game...and then some. The floor is open for questions, so let's dive right in.

(Tim) Was SimCity originally coded in Assembler or C or something else, and what problems did you incur in coding it? And are there any great hints you could give us?

WW: SimCity was originally coded in 6502 machine language for the C64.

BC: On the Amiga, it is mostly in C except for the screen stuff and the blitter stuff. There were some problems debugging it because of the multiple processes.

(Tim) Any hints?

WW: Don't cheat too much with 1.1.

BC: I have been trying to do BIG cities, and notice that you have to build slowly and wisely in order to have the density work. One tip is to build two zones adjacent with a rail or road on each side. That way there is always transportation to and from each zone.

(THol) Will, why don't Sims take trains unless compelled?

WW: The rail system is treated the same as the road system, except for traffic generation, pollution, and capacity. You will frequently get messages like "Sims want more roads", but you can ignore these if the system is all rails. It just means the Sims want these things, not [that they] need them.

(Jim W) Can I get a non-protected version of SimCity?

WW: Right now we have no [such] plans for the Amiga version. We are trying the Mac II version unprotected to see how it goes with sales; if it looks OK we might eventually start to unprotect all our stuff, but I can't guarantee anything.

(Jim W) Mine developed a read/write error, and I haven't had time to send in my registration (still here on my desk, even). Should I send in my disk with my registration card to get a replacement?

WW: Yes, if your disk has gone bad we'll replace it.

BC: Don't forget you can back your disk up and use the original as a key disk.

(Andy Hatchell) I loved your newsletter! In it, you mention that you will be uploading some City Data Files to bulletin boards. Does that include CIS, and, if so, where on CIS?

BC: Actually we encourage anyone to upload their City files.

WW: I think we will start putting some up on CIS.

BC: The Amiga version also will read Mac files. And vice-versa.

WW: Right, there are already several in the MACFUN SIG library.

(Steven Wartofsky) Will and Brian, a

double roads/rails question: 1) Why, when we completely rail in a part of the city, does it start to decline, and 2) Does building roads under rails provide any solution to transport problems?

BC: Take it, Will!

WW: I'm not sure why railing in a part of your city would cause it to decline unless it is not connected to the zones. And I don't think roads under rails would help anything, but it would take me a while to think that through. Odd.

(Tim) Was SimCity designed after any particular place? Your hometown maybe (originally I mean..)?

WW: Do you mean the concept? Or a [particular] city file?

(Tim) The concept.

WW: It was more designed around a Stanislaw Lem story.

(Tim) What story was that?

WW: I believe it was called "The Seventh Sally".

(Tim) Ah... just curious. Strange concept.

(Michael Sherrard) What enhancements are in the 1.1 version, and how much is the upgrade?

WW: Brian?

BC: Basically 1.1 takes care of a lot of bugs, mainly so that you "power users" can multitask and have cities going. We had a lot of memory eaten up by Chip RAM and have reduced that somewhat. For single-drive users, the file requester works properly. There is corner scrolling instead of jerky corner scrolling. When you save your City you can rename it at that point. The update is free if you send in your disk, $10 otherwise. Right Will?

WW: Right.

(John/TCR) Hi Will, Brian—Thanks for a great game! Hope Broderbund well appreciates your efforts! It's been a great seller for us here in Denver (35 copies sold and we're sold out again). I'm wondering what might be next on the list for Maxis/Broderbund. It wouldn't be a M.U.L.E. game, would it? (Wild stab!)

BC: No a H.O.R.S.E game. <Grin>

WW: I think our next releases will be SimCity PC and RoboSport Mac; as for our next Amiga game, that depends on what Brian wants to do.

AC: And what does Brian want to do?

BC: Hmmm. Let's spin the dice here and see what it will be. <Grin> Anything that pays. I'm not too particular.

(THol) Is there a strong likelihood of a Version 2 or SimCity Professional, or don't games work the same as paint programs?

WW: We have had so many inquiries from universities and planning departments that it looks like we will be doing a more serious version, but we're not sure of the timetable. Brian had mentioned doing a 512K version, but that's not decided yet. Brian?

BC: We need a 512K version for Europe. And I think one available for the States would be good too. Even though there are as many as 80% of the Amiga users with 1 meg, you can multitask better with a smaller game if you don't have the fat lady [the new 1 MEG Agnus] yet.

(THol) I'd really like ethnicity and faith so I can simulate Beirut. Priced at $159.95. What limits complexity?

BC: Errr, that's a better question for Will.

WW: I would say it's mostly processor speed; the memory is not so much of a problem. We're working on the next system simulation right now (don't ask me about it yet), and I am hitting a wall with the complexity issue running on a 68020. It will be refined to be faster, though.

BC: Time to drag out the '030 and the ol' Assembler, Will.

(Lee J) When combining road and rail systems, is it necessary to overlap road and rail squares to establish connectivity, or is it sufficient that the squares only be adjacent?

BC: My favorite part of the code (which I didn't write). <Grin> There are a few problems I'm aware of with connectivity which I hope to address in a future

version to make things connect up a little more logically. For now, just avoid the quirks. I would like to let users build freeways with roads parallel, for instance, instead of getting what you're getting now. Anyway, to answer your question, adjacent is good enough.

(Jim W) Brian, you mentioned the scrolling and the fatter Agnus earlier; how do you do the scroll? ScrollRaster()? ScrollVPort()? And will the fatter Agnus be implemented in future versions?

BC: The scrolls are accomplished by drawing all the tiles to the screen. The system scroll routines caused beam collision, and thus flashing, so we just redraw all the tiles in assembler without using the blitter (which is usually slower when doing 8x8). Fatter Agnus for now just lets you multitask more easily.

(Andy Hatchell) To cancel the scenario on a scenario city, you are supposed to save the city, then load it back. But what if you want to save a scenario and re-load it without cancelling the scenario?

BC: Scenarios are hard-coded; they can only be selected from the menu or startup at the beginning. They can also just be loaded in as a city. There is no need to save them back out as a city to just run them as a city.

(Andy Hatchell) Oh well. Getting through all 30 years of Dullsville in one sitting is tough!

BC: I see, Andy. Try it with a 68030. <Grin>

WW: Good point, Andy.

(Steven Wartofsky) Why, oh why, do planes crash even when one sets up the airports for 'em to crash over water, and why do they crash so frequently?

BC: Will, take it!

WW: We get this question a lot, which means it's my fault for allowing it to happen. The planes will randomly crash a certain amount, depending on the difficulty setting. But also if a plane or copter has a collision with something (other aircraft, tornado, Godzilla), then they will crash regardless of the difficulty. I think I should put in a collision avoidance routine to prevent this.

(David A.) It's possible for airports (and thus planes & helicopters) to be defined in 1900. Has this been changed in 1.1, or is it just part of the what-if capabilities of the simulation?

AC: (The same applies for nuclear power plants!)

WW: David, we had historical accuracy in one of the early Mac versions, but decided to take it out because it was rather confusing to people. ("I have enough money, why can't I build an airport NOW?").

(John/TCR) Can you release any sales figures (quantity) on the Amiga version of SimCity, and can you relate them to your sales of SimCity on other platforms?

WW: Yes, I feel it's OK to talk about it. As of yesterday we've shipped out about 8000 copies of Amiga SimCity to our distributor (Broderbund), which surprises me quite a bit. It's almost running neck and neck with our Mac sales (about 11,000).

(Frank Lazar) Have you considered SimCity 2100, Crisis at L-5?

WW: What I'm working on now has some similarities to that.

BC: Aha, a space station simulation! Sound like a good idea.

(Frank Lazar) No, it's an O'Neil Space city which gets minerals from the moon, energy from the sun, volatiles and supplies from Earth.

WW: Want a job as a designer?

(Frank Lazar) I only program in BASIC but want to learn. I like toying with concepts, though.

BC: You don't have to program to be a designer!

(Jim W) (Me, I'd like to see a BioSphere simulator). What about a multiplayer mode? Is that in the future, I hope?

WW: I think we are going to be pursuing stuff very much along those lines. There seems to be a void in the marketplace for interesting social/biological simulations such as SimCity, and we'd like to be the ones bringing this stuff to reality.

(THol) Is it a game, or a simulation? Is it (to you) just a bunch of simultaneous equations, or a statement about pro-car anti-rail? In brief, I'd like to hear what you think about the issues of making a more exact but less playable simulation or a simplified (maybe simplistic) but "fun" Lionel model train set.

BC: Will?

WW: I think the primary goal of this is to show people how intertwined such things can get. I'm not so concerned with predicting the future accurately as I am with showing which things have influence over which other things, sort of a chaos introduction, where the system is so complex that it can get very hard to predict the future ramifications of a decision or policy. Brian, anything to add?

BC: I think the more exact simulation would be a professional product as we discussed earlier. Something that would need to run on a more powerful system, though that could be an Amiga 3000.

<Grin> It should include waste disposal problems, educational zoning, etc., among the more detailed situations.

(THol) With respect to chaotic behavior, have you ever seen any "oscillating" cities which boom and bust over and over? Mine are monotonic.

WW: You will notice some periodicity in the employment cycle, but there are also many factors which will dampen the oscillations.

(Steven Wartofsky) Does city growth and/or decline become dependent at all upon an invisible external economy going through cycles (external market), or is it entirely the product of careful internal planning?

WW: The External Market was at first just a linear ramping function, but I didn't like the fact that it was out of the user's sphere of influence. So I attached it to the city score so a city with a high score will experience a faster growing external market than a city with a low score.

(Jim W) Connection to THol's question about games versus simulation: How would you compare SimCity to Populous, in terms of content and intent?

WW: I like Populous. I played for several hours last week. I sort of wish they went for a slightly more educational slant, but I like the idea of playing God and having a population to follow you.

BC: I haven't played Populous enough to get into the game. Just kinda toured around the edges. Neat game though.

(Andy Hatchell) Why does the Zone Evolution Chart differ so much from what one sees in the actual game? And why can't I view the ZEC from within the game (despite what the manual says)?

WW: Brian?

BC: I wish we'd had time to move those over to a chart that could have been included in the package like the Mac zone chart. People have mentioned the inconsistency and I really haven't had a lot of time to look into it, but Will did the Zone chart screen (passing the buck).

(Steven Wartofsky) Correct me if I'm wrong, but I believe that in the Amiga version if you load the ZEC chart first, then use the L-Amiga M/N keys to switch foreground and background keys, you can have the ZEC available?

BC: Only if you have fat Agnus.

(Steven Wartofsky) Brian, is that fat or fatter?

BC: Fatter, as CBM calls it.

AC: Actually, it will work with the original Agnus on a single drive system, or if you unplug your external drive before booting the disk.

(MikeM) Can SimCity be installed on a hard drive, and if so what type of copy protection is used? Key disk or a manual lookup thingie?

WW: Brian?

BC: Yes, it can be installed on a HD. I don't know why so many people had problems with HD installation on 1.0. It is key disk, so it will ask for your SimCity 1.0 or 1.1 disk (depending on what you've got).

AC: Okay, time to wind this one down; Will and Brian have done more than their duty tonight. Thanks again Will, Brian, and Maxis!!

BC: Rick, you're quite welcome! We enjoyed it!

WW: I just wanted to say that we're still a rather small software company right now, and we make a lot of silly mistakes from time to time. I just want to thank our customers for supporting us so that we can stay in business and attempt to raise the quality of entertainment software a bit.

(Steven Wartofsky) Will, there are much bigger companies that have been a lot less conscientious than you folks! Congratulations on doing such a good job.

WW: Bye all, and thanks for a great CO!

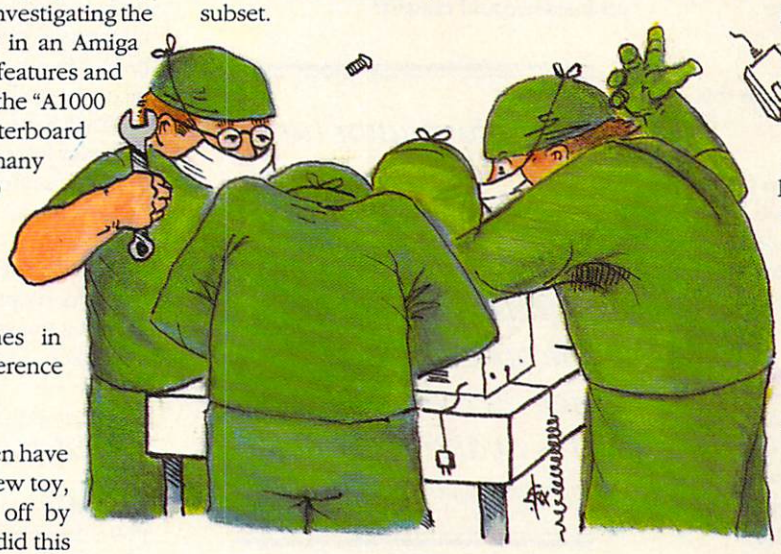
•AC•

Gregory Tibbs' A1000 Rejuvenator

edited by Richard Rae

On August 9th, 1989, CompuServe's AmigaForums hosted a formal conference with Gregory Tibbs, an electrical engineer at Wright-Patterson Air Force Base and president of Amiga-Dayton, an Ohio Amiga users group.

Shortly after receiving a 1 meg Agnus chip for his A2000, Greg began investigating the possibilities of installing it in an Amiga 1000. The concept grew in features and capabilities until it became the "A1000 Rejuvenator", a WCS daughterboard replacement which brings many of the features of the A500 and A2000 to the original Amiga model. The following is a heavily edited transcript of that conference. Names in parentheses represent conference participants.



AC: Greg, I don't think I even have to introduce you or your new toy, so I will just start things off by asking you if you REALLY did this just because you were told you couldn't!

GT: I am a person motivated by challenges. When things get old hat, I get bored. So I really DID do the Rejuvenator because it was said that it couldn't be easily done! It took tons of work; I started in mid-May with the design concept, and I now have three working prototypes and am working toward getting the design mass produced.

(Jim W.) Greg, to crush the rumors, just exactly what do we get for our (how much?) money?

GT: The Rejuvenator acts, in its minimal configuration, as a 1 megabyte RAM upgrade; this RAM is your new Chip RAM. Your [existing] motherboard RAM, through clever design, becomes Fast RAM if you use the Kickstart ROM, or 256K of Fast RAM and 256K [of] WCS. In any case, a 512K Amiga winds up with 1.5 meg (with ROM) or 1.25 meg (with Kickstart WCS). For good measure, you also get an A2000 compatible real time clock and an A2000-style video slot subset.

AC: Greg, any cost estimate at this point?

GT: It's really too early to tell. A lot depends on negotiations with CBM to get the 1 meg Agnus and ROM at less than dealer repair list price. I am shooting for $500, but I am not guaranteeing that I will make it.

(Jim W.) Will you still be able to get the 512K Fast RAM if you use a MultiStart like the one from Michigan Software?

GT: The Rejuvenator acts just like the ROM-based add-ons. If there is no physical incompatibility then I guess you could keep it, but you are going to have [an empty] ROM socket on the Rejuvenator.

One note. [The Rejuvenator] replaces the Kickstart daughterboard. CBM currently does not sell the WCS as a single repair parts item. You should be able to sell it to your dealer — assuming it is in good condition — or you could try to sell it to someone in a local user group. I feel it will be worth $50 to $60, as a good WCS is hard to get nowadays.

AC: Greg, you said "but you'd have a ROM socket...". Will the Rejuvenator let one buy a KS ROM, plug it into the new daughterboard, and get the basic functionality of, say, a KwikStart board (plus the additional RAM)?

GT: Yes. Since the WCS goes away, you do not recover that memory as it is no longer in the computer. You will have the KS ROM plus 1.5 megabytes of RAM (on a 512K machine).

(Jon Scarpelli) How will this affect those of us with Insiders or Spirit boards? Any incompatibilities? And what did you mean by video slot SUBset?

GT: I know that the Spirit board will physically fit. Other people who have seen the Insider say that it occupies less space than the Spirit board; if that is true, then it too should fit.

I refer to the A1000 Rejuvenator's video slot being a subset because there are some lines [missing which] have little to do with video; if I provided a method of obtaining them from the motherboard, [it would turn] the design [into] a kludge. [The] beauty is that [the current design]

requires no jumpers or trace cuts on the motherboard, and only three 'clip lead' jumpers from the Rejuvenator to the motherboard

In particular, the A2000 video slot has complete access to the parallel port. It also has provisions for dual audio in/out. I decided to not run these wires, as I felt that most A1000 owners would use it for the flickerFixer (which has been tested) and the A2300 genlock (which hasn't). It was an engineering trade-off.

AC: I think my system may be typical of many; I have two devices on the expansion bus. What with all the problems we've heard related to loading, how would the Rejuvenator fit into such a system? Since it replaces the daughterboard, is it safe to say "if it works now, it will work with the Rejuvenator", or not?

GT: I have three points to make on this subject:

1) I could have used a Gary chip and turned the A1000 into a real A500. I didn't because I wanted to use PALs to emulate the A1000 timings as closely as possible. Pals also meant one less chip to negotiate for.

2) Because the timings are as close as I can make them, most expansion devices should work. I am using better (i.e. faster 15 nsec) PALs and have more than enough copper on the power busses to ensure that there will be no noise problem. On my prototypes I see little noise over what is on the motherboard's power bus.

3) The one problem that could occur is bus loading of the 68000. There is somewhat more capacitance on the address and data bus, and a couple more tri-state devices to leak and load the bus. This might make a marginal situation terrible, but I usually recommend going to the 68010 CPU, as it has CMOS drivers that are more noise immune.

AC: What WON'T run with your video slot subset? Also, knowing how tight the 1000 is for space (especially with an Insider or Spirit installed), I have to ask: where does the video board GO???

GT: As I stated before, anything that uses the video slot's parallel port or audio mixing capability will not work. I haven't a list of what is available for the A2000 video slot, so I can't say what won't work.

The space situation is indeed tight. I intend to sell a new RF shield with the Rejuvenator that will add about .4" of clearance. You can also (optionally - you do not have to do this) trim the gold posts that the WCS mounts on by about .3". The only limit on how low you can go is a large electrolytic capacitor on the motherboard. The video card mounts horizontally, parallel to the Rejuvenator, between the power supply and the center case support.

AC: So, for example, a flickerFixer and an Insider could coexist?

The Rejuvenator turns your A1000 into a 1.5 megabyte machine that uses the 1 meg Agnus. You will have the option to add the new ECS Denise when it is available in the future.—GT

GT: Sure. The only thing I know for sure that conflicts is LUCAS' Frances RAM board. It will fit, with some height adjustment in LUCAS/Frances, but Frances will occupy the same space as the video slot card.

(Jon Scarpelli) Do you get the sense that CBM is reluctant because of prolonging what they may view as an inevitable conversion to the A2x00, or do they seem willing to listen/assist?

GT: I knew some form of this question was going to be asked. I don't think CBM knows WHAT to think about it. Anything more I can't say, as I don't want to jeopardize future negotiations with them.

(Willie Schreurs) Will there be support for the full ECS (Expanded (Enhanced?) Chip Set)? In other words, will Obese Agnus be joined by its similarly fattened brother and sister? Also, you earlier mentioned that the WCS goes. Does this mean no more ability to swap between Kickstarts on disk? <sob>
GT: If you have an ECS Denise beta you want to send me, I'll guarantee compatibility!

(Willie Schreurs) <Empty hands>

GT: Seriously, I will try to support the ECS Denise. It is supposed to be pin-for-pin compatible with the existing Denise. If so, then it should work.

As far as the Kickstart disk, have faith! I mentioned that one of the jumper options is to turn 256K of you motherboard RAM into a new Kickstart RAM. My next PCB artwork will incorporate the ability to switch back and forth via a mechanical switch between Kickstart RAM and ROM. The change will take place after the next keyboard reset of the Amiga.

(Pat Fallon) Glad to hear of a product with video option for the 1000! Can you hazard a guess when it might be available?

GT: I certainly want to hit the Christmas market. I am aiming for mid-November.

(Dean) How is the board installed? Are there connectors like on the WCS board or is it a slip over the posts and (maybe) solder?

GT: The board is larger than the original WCS. It extends over the three custom chip sockets, and the board has machined-pin adapters that fit in these sockets. The Paula and Denise chips plug into the Rejuvenator, along with the 1 meg Agnus. I had to have access to the entire data and address bus, so I needed all of the pins that were used by the WCS. I originally thought to solder those as I was concerned about noise. However, for this to be a commercial product, I can't have Joe Average Solderer mangling his A1000 beyond repair, so I use the same red slip-fit connectors as on the original WCS.

(Dean) Ok, so its just plug and go!

GT: Yes, along with the three clip leads to the motherboard.

(Daniel Habecker) My question has been half answered already. What about electrical compatibility with LUCAS/ Frances?

GT: I have a LUCAS board, but haven't tried it. Brad Fowles sent me a Frances board to check on the physical compatibility. Lucas is a little temperamental. It should work, but you might have to go through the problem of changing oscillators and/or 7474s again until you get it to work. The fact that I am using 15 nsec PALs will just about assure that you will have to 'recalibrate' it.

(Daniel Habecker) Thanks, I hope I can get it to work. It will give the 2500 a run for the money.

(Khalid) Going back to the video slot. Is it possible to have more than one video slot on your product (or for that matter the 2000)? Why/why not?

GT: I am looking at changing how the right angle connection is made. I may go to a header block pin/socket arrangement to a 2nd PCB to obtain the right angle. If I do that, aside from vertical space considerations inside the A1000, it could be possible. However, it depends on what video cards you wish to connect. Some of the signals are meant to go from the video card to the A1000/A2000 if you are using a genlock. This would lead to a contention problem that could damage the video cards. If the purpose was a genlock and a flickerFixer, then okay, they might coexist. I buffer the digital RGB signals, so loading them will not be a problem.

(J. Mikell) Greg, three quick questions: 1) Availability of unpopulated boards? 2) Cost thereof? 3) Speculations on Toaster compatibility?

GT: To work with the Toaster?!! Sure! Just ask for the $1500 A2000 add-on option! The Toaster is supposed to occupy several slots of the A2000 as well as the video slot. There will be no way of making it work.

Bare PCBs will not be available for several reasons that I am not allowed to talk about here.

AC: Who will be handling production and distribution?

GT: I am not affiliated with any company. I have a friend who is an Amiga dealer and has lots of money, and we are nearing an agreement. Don't worry. You'll find us.

AC: So you're not going through a distributor, you're doing it yourselves?

GT: It depends on the final production cost, which is now being estimated by an outside firm. I'm sure that it will be direct sales at first because we probably will be conservative on production quantities. When we get more confident and build thousands at a time, we will be able to lower the overall cost to afford a distribution layer.

(Charlie) I came in late, so I'm not sure I have a clear grasp of the thing itself; so could you briefly run down the major features/advantages? And mostly, has any provision been made for processor/co-processor options?

GT: [The Rejuvenator] turns your A1000 into a 1.5 megabyte machine that uses the 1 meg Agnus. You will have the option to add the new ECS Denise when it is available in the future. So, as long as CBM supports the A500 & A2000, this board which replaces your Kickstart RAM board will keep your A1000 current. It has an A2000-style video slot and a battery-backed real time clock. You can use the CBM KS ROM or, via a jumper, turn 512K of RAM into a new WCS (KS RAM). It will coexist with most major internal RAM expansions. And finally, the ROM socket is wired to accept the 512K byte ROM, should CBM ever decide to use one.

This is not a bus expansion product; it will not act like a coprocessor slot, and there is no provision for a math coprocessor.

(Caleb(BU) You mentioned a jumper for selection of KS ROM or RAM. Is this jumper internal, external, or a soft-switch?

GT: I have referred to it both ways. In my prototype, it is one of those square two pin jumper blocks. In the future, I will run it to an available flip-flop which will be clocked by the reset line, and have the flip-flop input a switch which will run outside of the case. To change, you'd toggle the switch and reboot via Ctrl-Amiga-Amiga. At that point the new setting would take hold.

(Caleb(BU) I expect that this will be a well-received product. As such, how small an initial production run do you expect to make? Are you thinking of doing something like ASDG and polling users first to see who will buy so as not to be under-stocked for too long?

GT: That depends more on finances. We plan to build 100-200 for a first run, depending on production costs. I am reasonably sure that they will go quickly. But we need to impress the bank!

(Caleb(BU) I think you will, Greg. The Rejuvenator sounds like a winner!

•AC•

Bug Bytes

The Bugs and Upgrades Column

by John Steiner

A posting on PeopleLink (thanks, Allegro) alerted me to a problem with MicroEmacs, the "other" freebie editor that is packaged with the Workbench on the Extras disk. After some experimentation, I have verified the bug, although I don't really have any workarounds. It seems MicroEmacs does not clear the archive bit when it saves a file.

This bug could have disastrous consequences for hard drive users that perform incremental backups. Hard disk backup utilities, like Quarterback, automatically set the archive bit when they finish backing up a file. And programs that write to previously existing files should clear the archive bit whenever they save a file.

Backup programs use the archive bit to determine if a file should be backed up when choosing an incremental backup. Incremental backups do not back up the entire drive, but do back up those files that have changed since the last backup session. If you are using MicroEmacs on a hard disk, caution is certainly in order when doing incremental backups.

While on the topic of Commodore products, there has been a smattering of postings on the networks regarding some hardware incompatibilities with various brands of third-party expansion products and the revision 6 motherboard (revision 6 boards are being shipped with the new Fatter Agnus chips in the 2000's and 2500's.) If you have a revision 6 motherboard and have experienced problems with third-party products, let me know. If any manufacturers have provided details on specific problems and/or workarounds, let me know, and I will pass along the information.

A bug in Music-X that caused the serial port to remain allocated after the program executed has been fixed. Registered owners can contact the folks at MicroIllusions for details on the upgrade.

HiSoft BASIC Professional from MichTron has been upgraded to version 1.05. You can upgrade your copy by sending your original disks and $5.00 to cover postage and handling to MichTron.

According to a press release from Gold Disk, another major upgrade for Professional Page is about to be released. It should be available by the time you read this, if the September release date is met. The upgrade features CompuGraphic fonts in addition to the large number of Adobe Postscript fonts. Along with several major features mentioned in the press release, the program will include high-quality, dot-matrix and non-Postscript laser printer support—a feature currently missing from this otherwise excellent desktop publishing package.

X-CAD, probably the most powerful and most complicated Amiga CAD program, has problems with the dongle and the A2500. The program does not recognize the presence of the dongle. Haitex Resources, the people who have the licensing rights to market the program, have been providing corrected disks to A2500 owners who have the original version of X-CAD.

But when one user received his upgrade, he found no indication of better performance. He initially attempted to simply drag the icon from the new floppy disk onto his hard drive, overwriting the old version.

A quick call to technical support provided a solution (no documentation was included with the upgraded disks). Before running the program, delete the old version from your hard disk entirely, and install the new version from the beginning. That should solve any dongle problems.

Another tidbit of information was made available: it is possible to get a non-donglized version of X-CAD from Haitex, along with a disk full of support programs and utilities. The cost of the upgrade is $29.95, and be sure to include your original master X-CAD program disk. For more information on the upgrade, contact Haitex Resources.

CADVision International, original creators of X-CAD, has not been happy with the sluggish sales of X-CAD in the United States. The program has a reputation of being difficult to learn, with little useful documentation being provided. The program currently requires a minimum of 2 MB of RAM, further limiting its potential sales to those equipped with lots of memory expansion. As a result, CADVision has completely redesigned the interface, and is now planning to market two new versions of the program—X-CAD Designer and X-CAD Professional.

X-CAD Designer is a step-down from the original X-CAD, yet still retains many of the features and high-speed capabilities of the original X-CAD. X-CAD Designer will run on any Amiga with 1

MB of RAM or more. The program also boasts a much lower price of $149.95. A press release details several features including high-resolution graphic support for 24-pin and laser printers, optional import/export of AutoCAD DXF format files, IFF file export, and Aegis Draw Plus import. The program is also compatible with Gold Disk's Professional Page. X-CAD Designer's big brother, X-CAD Professional, will essentially replace the current version of X-CAD.

Neither program will be marketed through Haitex Resources. Instead, distribution will be through one of the largest wholesale Amiga software distributors. Since Haitex is no longer marketing the new versions, there are some technical support questions involved. The U.S. software distributor does not have technical support facilities, and the only phone number listed for CADVision International is in Great Britain.

The person I spoke with at Haitex did not know which, if any, upgrade paths would be available to current X-CAD owners who wish to move to the soon-to-be-released X-CAD Professional.

ASDG Incorporated has released version 1.1 of the Professional ScanLab software. Professional ScanLab software drives the Sharp JX-300 and JX-450 color image scanners.

Due to improved 24-bit electronic color separation routines, the upgrade sports noticeably truer colors than the earlier version . The upgrade eliminates problems with color shifts caused by impurities in the inks used in the final printing process.

For more information and details on upgrading Professional Scanlab, contact ASDG Incorporated.

Gramma Software has announced an update to NAG PLUS 3.1. Several features and improvements include the addition of an ARexx port, more keyboard command equivalents, unassisted phone dialing via modem, and several bug fixes. Registered users may receive an upgrade through an order form distributed by Gramma Software, or by contacting the company directly. Cost for the upgrade is $10.00; a disk and bound manual supplement may be ordered for $20.00.

SimCity, from Maxis Software has a bug-fixed upgrade. The new version is less hungry for Chip memory, and is easier to load and run. Several bugs have been squashed as well. Registered owners can call for upgrade information. If you are a registered user and have not received an upgrade, contact Maxis Software.

ProWrite version 2.5 is now available from New Horizons Software. Improvements to the program include better printing routines, user-adjustable page sizes, and faster, interactive spell checking. The upgrade costs $20.00 plus $5.00 shipping and handling. Send your original ProWrite program disk and payment to New Horizons Software.

That's all for this month. If you have any workarounds or bugs to report, or if you know of any upgrades to commercial software, you may notify me by writing to:

John Steiner
c/o Amazing Computing
Box 869
Fall River, MA 02722

or leave EMail to Publisher on People Link or 73075,1735 on CompuServe.

•AC•

Companies Mentioned

MicroIllusions
17408 Chatsworth St
Granada Hills, CA 91344
(818) 360-3715
Inquiry #210

MichTron
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700
Inquiry #211

Gold Disk, Inc.
Box 789
Streetsville, Mississauga ON
Canada L5M 2C2
(800) 387-8192
Inquiry #212

Haitex Resources
208 Carrollton Park, Suite 1207
Carrollton, TX 75006
(214) 241-8030
Inquiry #213

CADVision International
1612169 Uxbridge Road
London, W13 9AU England
011 44 (1) 603-3313
Inquiry #214

ASDG Incorporated
925 Stewart Street
Madison, WI 53713
(608) 273-6585
Inquiry #215

Gramma Software
17730 15th Ave N.E., Suite 223
Seattle, WA 98155
(206) 363-6417
Inquiry #216

Maxis Software
953 Mountain View Drive Suite 113
Lafayette, CA 94549
(415) 376-6434
Inquiry #217

New Horizons Software, Inc.
Attn: ProWrite Upgrade
PO Box 43167
Austin, Texas 78745
(512) 328-6650
Inquiry #218

(continued from page 18)

shortened as gameplay progesses, thus speeding action. The number is then displayed at the top of the screen using the LOCATE command. A note is played based on the y coordinate of the falling letter. This continues until the letter hits the city, or the correct key is pressed.

A handy routine I frequently employ is the "key.wait" subroutine. Whenever I need to wait for a key press, I call this routine. It displays the message "Hit any key to continue", then waits for a key press. Once a key is pressed, the text is removed by printing spaces over it, and the program returns to where it was called from. It might be worth setting up this routine as a subprogram and passing to it the text to display, and the coordinates of where to display it.

Typing Tutor Listing

```
init.some.stuff:
  DIM scores(11),names$(11)
  RANDOMIZE TIMER
  WINDOW 1,"Typer Tutor    1989 Mike Morrison. Written for
Amazing Computing",(0,0)-(631,186),10

init.hi.score:
  x=0
  ON ERROR GOTO handle.error
  OPEN "letters.score" FOR INPUT AS 1
  WHILE NOT EOF(1)
    x=x+1
    INPUT #1,names$(x),scores(x)
  WEND
  CLOSE #1
  hscore=scores(1)
  GOTO main

handle.error:
  IF ERR<>53 THEN ON ERROR GOTO 0
  FOR x=1 TO 10
    names$(x)="Empty for now"
  NEXT


main:
  del=500:lives=3:score=0
  GOSUB init.screen
  GOSUB get.ready

  WHILE lives>0
    GOSUB get.char
    GOSUB update.score
  WEND
  LOCATE 10,35
  PRINT "GAME OVER"
  GOSUB key.wait
  IF score>scores(10) THEN
    GOSUB hi.score
  END IF
  GOSUB play.again
GOTO main

init.screen:
  COLOR 1,2
  LINE(0,7)-(631,176),2,b
  PAINT (1,1),2:LOCATE 1,20:PRINT "Hi-Score:";hscore
  LOCATE 1,40:PRINT "Lives:";lives
  LOCATE 1,60:PRINT "Score:";score:COLOR 1,0
  FOR x=0 TO 631 STEP 8
  h=0
    WHILE h<7
      h=INT(RND(1)*20)+1
    WEND
    LINE (x,176)-(x+6,176-h),2,bf
  NEXT x
RETURN

get.ready:
  LOCATE 15,26:PRINT "Move the mouse out of the way."
  GOSUB key.wait
  LOCATE 15,26:PRINT "                                "
RETURN

play.again:
  LINE(222,67)-(386,84),2,bf
  LOCATE 10,30:PRINT "Play again (n=END)"
  r$=""
  WHILE r$=""
    r$=INKEY$
  WEND
  IF UCASE$(r$)="N" THEN
    CLS
    COLOR 2,1
    LOCATE 2,33:PRINT "Keyboard Aces"
    COLOR 1,0
    LINE (128,32)-(464,144),2,bf
    COLOR 1,2
    FOR x=1 TO 10
      LOCATE 6+x,20:PRINT names$(x)
      LOCATE 6+x,50:PRINT scores(x)
    NEXT x
    COLOR 1,0:LOCATE 20,33
    PRINT "See you later.";
    END
  END IF
  CLS
RETURN


get.char:
  xc=INT(RND(1)*77)+1
  c$=CHR$(INT(RND(1)*26)+65)
  y=1:a$=""
  WHILE a$<>c$ AND y<22
    FOR q=1 TO del+spd:NEXT
    SOUND 400-y*10,1
    a$=UCASE$(INKEY$)
    y=y+1
    LOCATE y,xc:PRINT c$
  WEND
RETURN

update.score:
LINE ((xc-1)*8,8)-((xc-1)*8+7,7+((y-1)*8)),0,bf
  IF a$=c$ THEN
    score=score+(10*(22-y))
    IF score>hscore THEN hscore=score
  ELSE
    lives=lives-1:spd=0
    del=500-(100*(3-lives))
    BEEP
  END IF
  COLOR 1,2
  LOCATE 1,20:PRINT "Hi-Score:";hscore
  LOCATE 1,40:PRINT "Lives:";lives
  LOCATE 1,60:PRINT "Score:";score
  COLOR 1,0
  spd=spd-10
RETURN

hi.score:
  CLS:here=0
  COLOR 2,1
  LOCATE 2,33:PRINT "Keyboard Aces"
  COLOR 1,0
  FOR x=10 TO 1 STEP -1
    IF score>scores(x) THEN
      scores(x+1)=scores(x):names$(x+1)=names$(x)
      scores(x)=score:names$(x)="":here=x
    END IF
```


(continued on page 108)

Roomers

by The Bandito

[The statements and projections presented in "Roomers" are rumors in the purest sense. The bits of information are gathered by a third party source from whispers inside the industry. At press time, they remain unconfirmed and are printed for entertainment value only. Accordingly, the staff and associates of* Amazing Computing™ *cannot be held responsible for the reports made in this column.]

AmiEXPO Chicago Report

The usual companies were present. Not surprising was the absence of MicroIllusions and Aegis; both of them have been avoiding shows lately because of the expense. Some interesting hardware was announced: a series of transputer boards for the A2000—how does 8192 x 8192 pixel resolution with 16 million colors grab you? Of course, a mere 20 megabytes of video RAM will be needed to use it. No telling about the price, but it would probably be enough to make a Mac II owner take notice. How's that for high-end? For the old faithful A1000 owner, the Rejuvenator sounds like it will put some life into the old 1000 series by offering a way to upgrade to the new Enhanced Chip Set and providing an A2000-compatible video slot.

Commodore has continued its hiring blitz by adding a new VP of marketing, Lloyd Mahaffey, who was previously in charge of federal sales for Apple. Not surprisingly, the Bandito has heard that Commodore plans a big push into the federal market. Some major defense contractors already use the Amiga for presentations, including such well-known names as Martin Marietta and Lockheed. While the Amiga may never make much headway against the installed base of PC's, it is possible that the Amiga could be the computer of choice for the interactive video, multimedia, and presentation requirements of the Feds. After all, the Amiga is about half the price of comparably equipped Macintosh systems, and government contracts are sensitive to such details.

Commodore is starting to get better coverage in mainstream publications like *PC Week*. A recent article has corporate users describing why they prefer the Amiga for presentations. As Commodore hires more marketing people to handle specific vertical markets, expect more coverage of the Amiga in specialty magazines for video, graphic design, and publishing.

The Bandito has heard that George Lucas is producing a commercial for Commodore, to be directed by Matthew Robbins. Apparently George is a fan of the Amiga and offered to do it for less than would be expected.

Commodore stock has taken a beating lately, dropping from 18 down to around 10 because of the poor quarterly returns (a $9 million loss in the latest quarter). Commodore sales were down to $180 million from $215 million; they lost $8.9 million versus a $12.2 million profit in the same quarter last year. Sales have slowed in West Germany, and the stronger dollar has also hurt overseas profits. Commodore has also had more expenses lately, because Copperman is hiring people from Apple, trying to set up a really thorough developer support group and a well-staffed marketing department. Maybe it's time to buy the stock.

Commodore is gearing up for a heavy Christmas advertising schedule, while Apple is cutting back because of poor quarterly results. The rumored budget for Commodore's advertising is $14 million, which would make quite a splash. It sounds like Copperman wants the Amiga to hit big this Christmas and get some name recognition (as well as sales).

The Bandito has heard that George Lucas is producing a commercial for Commodore, to be directed by Matthew Robbins. Apparently George is a fan of the Amiga and offered to do it for much less than would be expected. If this is really true, the commercial should cause quite a bit of attention, since it will be the first that Lucas has had a hand in. The Bandito is working hard to find out just what the commercial will be like; watch out for further details.

Say, has anybody seen that Desktop Media advertising campaign touting Macintoshes? It offers some amazing animations of a flying car, which zips along at about 3 frames a second and looks like it was drawn by a 4-year-old (no offense, kids). It should convince every animation professional to buy an Amiga! Commodore should try to buy time right afterwards so people can see the incredible difference between the cheesy looking Macintosh Helocar and, say, any Sculpt 4D demo. Hey, a new slogan for Apple—"Half the power at twice the price!"

Connectivity is becoming a more important buzzword at Commodore. Expect to see their Novell network hookup on the market soon; it's already been demoed at shows. Commodore may cut some deals for other types of network support. There is already a couple of solutions on the market: Ameristar has an Ethernet card for the A2000, and CMI has an Appletalk card. All they need is major software support, and perhaps the benefit (to the corporate mind) of a Commodore label on some of these network solutions. The Bandito hears that a lot more Amigas would sell if people could hook them up to their network at the office.

I'll Gripe If I Want To Dept

The Bandito hates copy protection almost as much as software piracy. Isn't it scary to insert a disk and VirusX says that it has a non-standard boot block? Then you realize it is just copy protection—or you hope it is just copy protection. The worst sort of copy protection is the kind that makes you look up a word in the manual, so then you have to keep the manual handy and try to count lines and words. A real pain, especially on a piece of productivity software. It's almost as obnoxious as a key disk protection set-up. Fortunately, more developers have been coming to their senses and getting rid of copy protection, but there is still enough out there to be annoying. The Bandito realizes that something has to be done to protect games, but there's no excuse for copy protection on a piece of software people depend on for their livelihood. What's the answer? For the users, boycott copy-protected software and make sure the manufacturer knows why it's not being bought. At the same time, try to stamp out piracy by refusing to indulge in it, and reporting BBS's and other heavy offenders to the publishers concerned. For the publishers, leave the copy protection out, but make sure the software provides a lot of value (especially a good manual and good technical support) at a reasonable price.

Interfaces that are so far from normal that you think you're using an Atari ST or a UNIX machine or something. Sometimes it is due to a rush job of porting a product from another computer system. Other times, the programmer just gets slap-happy and starts throwing buttons around like crazy. The Bandito's advice: Color-blind people shouldn't do interfaces. It would be nice if you didn't have to read a manual just to learn each new variation on a file requester. Maybe 1.4 will help this mess.

The Latest Word Is Perfect Dept.

After WordPerfect Corporation announced that they would no longer develop software for the Amiga, loyal Amigans raised a great hue and cry. With many companies, that wouldn't make a difference, but WordPerfect has gotten to the top of the heap by listening to its customers. So WordPerfect has bowed to public pressure and decided to continue working on WP software for the Amiga. It's really a non-announcement, because they were going to do that anyway—at least to the extent of supporting the current version, bug fixes, and a rev for Kickstart 1.4. Still no plans for a WP 6.0 on the Amiga, though once the IBM version is done they may consider it. If you want graphics and non-embedded formatting, you will have to use some other Amiga word processor like *Excellence!* The Bandito has heard that for those who just want to put words in as fast as possible, *Transcript* from Gold Disk is the best solution.

About Kickstart 1.4

Expect new versions of all major software packages. Why? Well, most will want to take advantage of the new graphics modes allowed by the Enhanced Chip Set and 1.4, so they have to rev the software. And though we were promised by Commodore that our software will work properly if it followed all the rules, you can bet that a lot of programs will not work right under 1.4. So start saving your pennies for the inevitable slew of upgrades. Hopefully, the upgrades will offer more than just working with 1.4, and we'll get more features for our money. We'll see.

What's hot: Sim City, Populous, Licence to Kill. What's not: ports of old C64 software. Things are pretty dull, really, in the Amiga market lately. Where is a hot new productivity product to get excited over? The Bandito's hoping that Christmas will prove more interesting. At least a new crop of games will be out. There's some excitement over HyperClone software, but the Bandito just can't get too thrilled about something that's really a programming language. No matter how you slice it, that sounds too much like work. HyperClone Wars are coming this Christmas. The Bandito expects the battle to be waged in advertising, on networks and bulletin boards, and through user groups. ULTRACARD has an early start, but there is no marketing to speak of. The competition should be arriving this fall, and we'll see some competitive fireworks. The big question is how the HyperClones stack up against each other, or rather who gets the most stacks out into the market. After all, it's the stacks that really do the work.

The HAM Paint Wars are raging again as Digi-Paint 3 finally ships. Let's see, what's the status of the combatants? The Bandito hears that Digi-Paint 3 is a hot seller, especially since people have been waiting to see it. Deluxe PhotoLab is silent, as is Photon Paint 2.0. Are they out of advertising ammo, or are there deeper reasons at work? DeluxePaint III is still humming along with a bit of advertising but not making too much noise. A new version of DeluxePaint III with an extra feature or two and a number of bug fixes is out now, though there's no formal upgrade (call EA's customer service to get a copy). What's confusing to the Bandito is that DeluxePaint II is still being offered, now at $99.95. Are there many takers? The Bandito hears that most are opting for DeluxePaint III instead. At stake in the HAM Paint Wars is the lucrative Christmas sales market. The Bandito's tip: bet on the big ad dollars to win.

Hey, with RAM prices dropping back to where they were a couple of

years ago (and lower), it's time to consider making the A500 with 1 megabyte of memory standard. Maybe the A2000 should have 2 megabytes standard. Multitasking only really gets useful when there is at least 3 megabytes. More programs are now requiring at least 1.5 megabytes. The Bandito's looking forward to seeing the first game that requires 3 megabytes. It should be simply amazing, don't you think?

Speaking of prices headed earthward, color printer prices are dropping as their sales improve. You can get an HP PaintJet for under a thousand, these days. The Bandito hears that prices of color inkjets will hit $800 on the street by year-end. Programs like PenPal look even more fun. Being able to include a digitized color image in a letter can make a powerful impact. Think of the love letters you can write...

Some of the old Amiga developers that have been in deep financial waters lately are about to be sucked under the rising tide of debt. The theory is that if they can survive till business picks up in the fall, then maybe the rising tide that lifts all boats will pull their ship out of the mud. That is as long as they've managed to plug the holes in the hull. Meanwhile, the rats continue to send out their résumés.

The Japanese are busy cooking up some interesting hardware. The latest developments: Fujitsu is making the FM TOWNS, a 386 PC with a CD player built-in, eight-channel hi-fi stereo, and a software interface to the CD player and the stereo (click Play to play a CD, for instance). Hitachi has a combo PC, telephone, and word processor called the Proset 30 with a built-in color monitor, printer, internal hard disk and modem. Sure, these sound like weird combinations, but they just might hit on something. Wouldn't you like a stereo receiver on a card for your A2000?

Byte-by-Byte is bringing out a version of Sculpt 4D for the Macintosh II. Well, it won't have any animation or surface mapping in the first version, so it's not as powerful as the Amiga version. Oh, yes, the price tag is a bit different — would you believe $1500? Expect to pay $1000 for each animation or surface mapping module. According to the Bandito's pocket calculator, it's cheaper to buy an Amiga 500 and a copy of Sculpt 4D than it is to buy the Macintosh version of the software. Are Mac buyers really that unconscious? Ask Byte-by-Byte in about six months.

Hard disks are getting more popular in the Amiga market, since the price is now comparable to an IBM or a Mac hard drive. If you're doing serious work, it's indispensable. Now if only more software developers would make it easy to install their software on a hard disk, the Bandito would be much happier. Let's start with Commodore—their hard disk formatting software seems to be written in CP/M by a learning-disabled chimpanzee; it was documented by his twin brother.

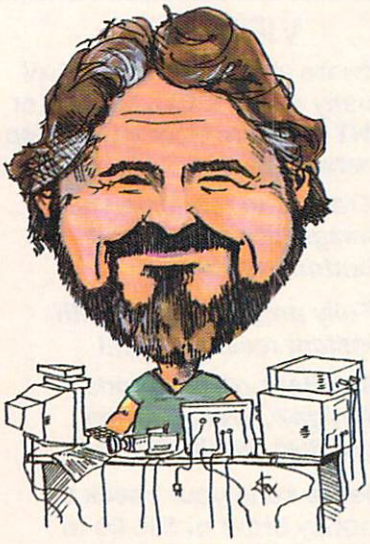
The more the Bandito hears about 1.4, the more it sounds like the Macintosh operating system. Hey, that's not bad, because they've got a lot of good ideas. Of course, the Macpeople haven't figured out useful things that we already have, like multitasking or a command line interface option. Give them a few decades, and maybe they'll clue in. The most disappointing thing about the 1.4 alpha is the graphics. Supposedly, Commodore hired a real graphics artist to design new icons and such, but it still looks rather sophomoric to the Bandito's critical eye. C'mon, let's get a real professional looking font. And it has to be readable in the new 640 x 480 mode. One thing the Bandito does miss about the old 1000 is the cheery "bee-dee-bleeuroop" that would greet you when it was powered up. There should be some cool standard sound when you start up, perhaps a digitized "Welcome to Amiga!" Of course, it would have to be easy for the user to substitute their own start-up sound. Yeah, sure, if you want to hack around with your startup sequence you can do that now, but it should be easy and part of the operating system. While they're at it, how about making it easy to put in any picture as a backdrop for the Workbench?

The Bandito is a little worried about 1.4. Sure, it's great that Commodore is taking suggestions from all the developers about what to put into it, and 1.4 should be a great advance in many ways. But if Commodore ever wants to ship the bloody thing, they've got to draw a line and say "That's All, Folks!" Get something out for Christmas and put the rest of the stuff in 1.5. Let's get it out early and make Apple look silly trying to put out their new System 7.0 software with half the features in 4 times the memory in twice the time.

All is not well with the 1 megabyte Agnus chips. Some software refuses to run, particularly some games. Other games run extra slow, which is rather exasperating to those who got the 1 meg Agnus expecting better performance. There will be some upgrades, but don't expect all games to provide an upgrade.

Commodore can take a lesson from Apple on how not to introduce new models of their computers. Apple stock is wilting and their profits are slumping because they have introduced new models at a price that makes them much more attractive than the older computers in their product line. So, of course, people have stopped buying the older computers and are demanding the new ones; equally, Apple is still producing too many of the old computers and not enough of the new ones. While the Bandito would dearly love to see the Amiga 3000 on the market, Commodore should consider carefully how to introduce it, and reprice the product line so they do not run into similar problems.

•AC•



video Schmideo

by Barry Solomon

I was working on my Amiga last night, very tired after seven straight days of working (no rest for the weary!) when my drained brain insisted that I get horizontal and turn on the other tube. Yes, I admit it, my pre-bedtime ritual includes zoning out in front of the TV. Oh, I tell myself it's a great way to keep om top of all the latest professional video effects and styles, but I'm a child of the fifties and I just can't sleep without my fix. Anyway, there I was flipping through the channels when what did I spy with my little eyes? My animation. Yep, right there on my TV—my very first professional project!

It's not that I hadn't seen it on TV before. I had—but only once, and that was months ago in a neighboring town. Anyhow, seeing the piece got me thinking back to the trials and tribulations of my first professional video gig.

About two months after I had gotten my Amiga, I was still learning the most basic graphics and animation. I had started from scratch with no computer background whatsoever. But all of a sudden, I was in business. It seems my brother-in-law (in the video business) had mentioned my Amiga and my plans to a friend (also in the video business) who mentioned me to a friend (you guessed it.)

This friend was a very nice woman who, with two other very nice women, owned a local real estate company. She also produced a half-hour show that appeared daily on the local cable channel. It showcased their nicest properties and, from what she told me, it worked quite nicely for them.

The problem was that the show was on very frequently and, although the houses changed every week, she felt that viewers were becoming a little blasé. She wanted something new to grab their attention. A snazzy animated intro, perhaps? So she called me. Well, I made an appointment with her for that afternoon and immediately called in sick to my real job. (That darn cough!)

She was a little vague about what she wanted (like the Grand Canyon is a little crack), but she knew she wanted something new and different. I assured her that I could not only handle this job, but I could increase her viewership considerably. I told her I would call her in two days with the plan. All the way home I looked for a phone booth to change in. I mean, if I didn't turn into Mr. Video soon, I was sunk!

When I got home I immediately put on my thinking cap, which was no mean trick. It had been a while since I had worn it and I had gotten a little flabby around the brain. Luckily, my batteries weren't completely dead and, as I stared at the real estate company logo, it all materialized, crystal clear, in my head.

The logo consisted of a drawing of the side of a house with two film reels on one side of it. "How's about we animate those reels?" I said to myself. "Yeah, that's the ticket! We'll animate those reels and have the title of the show kinda FLASH in on the other side of the screen."

With my confidence steadfastly rooted in God-knows-what, I met with the woman two days later. My enthusiasm must have counted for something, because with absolutely nothing concrete to show her, she approved the idea. I was flying! My mind was racing with visions of fame and fortune when suddenly...

"But...it doesn't really show that it's video," she pointed out. "How about something with a TV?"

I swear I'll never leave my thinking cap at home again. Luckily, synapses jumped into full gear and I had it!

"We'll start just as I said. House and reels appear; the reels are turning; the title of the show pops in, word-by-word, and after about five seconds, we'll have a giant TV appear beneath the house with static on the screen which clears to black into which the cable company can key the first house of each show."

Being the sport that I am, I even offered to give them a separate 30 seconds of the animated logo with the weekly schedule on it, over which they could run narration and use as a promo. Well, she loved it! Hands were shaken (so was I), and I told her I would have it for her in 30 days.

Back at my home/office/studio, my adrenaline faded along with my hopes. I realized that, although I was sure it could

be done, I really had no idea how I was going to do this. But I realized it was sink or swim, and I really hate getting water in my ears.

The first thing I had to do was get the logo on my Amiga. No problem; I used my handy-dandy digitizer. My client had even given me camera-ready art. The digitized logo didn't come out so well, but I figured I could clean it up in my paint program. An hour later, with her logo looking more like a moose than a house, I knew I was in trouble. It seems the digitizer doesn't take too well to very thin lines. If only I had a drawing tablet, I could trace it right in. But that was clearly not in my budget. What to do, what to do?!!

I ran to my local stationery store and bought a couple of those clear plastic folders—you know, the kind you used to buy to put your book reports in to impress the teacher? Dashing home I placed this over my victim, the logo, and traced away with a fine point permanent marker. In fifteen minutes I had it captured forever on plastic. Opening my paint program, I drew a straight line across my screen about where I thought the bottom of the house should go. I then carefully held the plastic (with the captive soul of the logo) against my monitor, lined it up as best I could, and taped the sucker right there. In twenty minutes I had that logo right there on my screen!

I decided that my next step would be to animate the reels. Again, I figured no problem; I'll create the reels with their "spokes," cut them out as brushes, and use the ROTATE BRUSH feature of my paint program to create six different positions for the reels (actually, six positions for the large reel and four for the small reel, as reels of different sizes would turn at different rates).

Those of you who realize my glaring error in logic may skip to the next class. On attempting this step, I soon realized that rotating a circle as a brush does not give you a perfect circle—at least not with any of the three programs I tried. (Well, that explained the Flintstone-like wheels I had seen on so many car animations!)

Okay, if I couldn't turn the reels, I would move the spokes. Stamping six copies of the large reel and four copies of the small one on a blank page, I set about drawing the spokes. The little reel did actually have spokes, so that was easy. The large reel, however, resembled a standard large film reel, with circular holes around the center, through which you could see the film.

On this one I had to get clever. On a plain, solid circle I stamped six smaller circles around the center (representing the holes). Setting my paint program to BLEND, I then centered and stamped a circle in black (representing the film). The circle was blended over the existing circles, so you could see right through it. Then I simply switched back to regular paint mode and used my fill tool to re-color the reel and film parts.

Doing the TV part at the bottom of the screen was actually pretty easy, as was doing the promo I had promised. If you ever get a chance to do any kind of intro or titling that will appear on TV, you might want to keep this promo idea in mind. You can probably use the work you've already done and just plug in the air times as I did. Five minutes and I had something I could offer to my client as a FREE bonus.

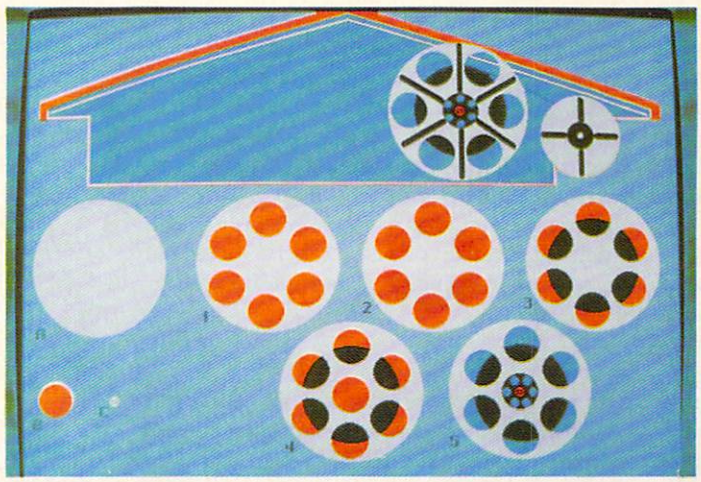
There are three other things I had burned into my memory banks by this project that you should also keep in mind:

1. Keep notes. Always use the coordinates option in your paint program. Keep track of where your objects are in the screen. Make special note of where you stamp brushes; you never know when you may have to duplicate your work. Also, if animating, it will be helpful to know where your brushes go (or don't go) on other pages.

2. Save your work (often!). I know we read this all the time and we all think it won't happen to us. It can. It does! I lost this entire project three times and had to begin again each time from scratch.

3. Save all your brushes to disk. Even things you are sure you will never use again. You may have to redo just that one page and it could be very difficult (or impossible) to re-cut a previously stamped brush.

Well, that's about it for this go-round. I'll be back again with more hints and tips for you. And don't forget, if you have any questions about the practical side of graphics for video, please write. If you want to know the know the horizontal sync rate for S-VHS don't write. I know, but I'm not telling!



The large reel was made from 3 brushes (A, B, C). The problem was having the film (or tape) show through the holes in the reel. I made a circle in low mix mode (using Deluxe PhotoLab) over the reel with the holes (2). This gave me a perfect circle outline. The proper colors were then filled in (3) and center holes and nut were added (4 & 5). The spokes were added on the second animation to exaggerate reel motion (final). With the spokes in place, only 3 reel positions were used to create the animation.

Barry Solomon, Video Editor
c/o Amazing Computing
P.O. Box 869
Fall River, MA 02720

•AC•

Fractals Part III

Saving 16-Color Pictures

BASIC and True BASIC programs that let your work with fractals in 16 colors and save the images to disk

by Paul Castonguay

In parts I and II we drew fractals using only four colors, because it was important in those first introductory examples to keep the programs as simple and as short as possible. But now we want to learn to do more on our Amigas.

Sixteen colors

The Amiga is advertised as being able to display sixteen colors simultaneously in high-resolution graphics mode (640 x 200 pixels). So where are they? Colors on the Amiga are linked to a concept called screens.

Screens

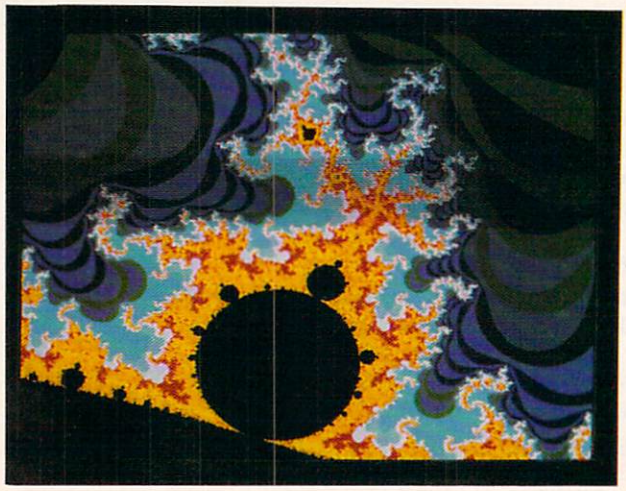
A screen is a display area that can have its own resolution (high or low) and a number of colors (2, 4, 8, 16, or 32). The Workbench screen you see when you boot up your computer is a 4-color, high-resolution screen. To display graphics in 16 colors, you must open a 16-color screen, done in AmigaBASIC with the SCREEN command. The Amiga is a multitasking machine, so when you open a new screen it runs simultaneously with the Workbench screen. You can view a screen either individually—by "stacking" it on top of all the others—or simultaneously with other screens—by shifting their positions and allowing partial viewing of each.

Windows

Normally we do not draw graphics directly onto a screen. To draw graphics—or even to print text—we must open another kind of display area on the Amiga, called a window.

A window display area is subordinate to a screen in two respects. First, a window must remain within the area of the screen in which it is opened. Second, a window must use the resolution and color properties of that screen. For example, if you open a window in the Workbench screen, it inherits 640 x 200 pixel resolution and four colors. To draw graphics or text using more colors, you must first open a new screen that has the number of colors you want, and then open a window within that screen.

This month's example:
Fractal generated on the Amiga and modeled after a picture in The Beauty of Fractals *by Peitgen and Richter.*



Let's try a little interactive experimentation...

Fire up AmigaBASIC and left-select (with the mouse) anywhere in the OUTPUT window to make it active. If you enter AmigaBASIC commands in this window they will be executed immediately. (Use single-line commands only.) This is called "immediate mode" on most computers. You type the command and the computer immediately carries out your request. Try it by entering

```
FOR I=1 TO 1000: PRINT I: NEXT I          [RETURN]
```

Your computer immediately starts counting to 1000. If you're impatient, hit [CTRL]-C to stop it. This termination is possible because AmigaBASIC is an interactive language. Now enter a more interesting command:

```
SCREEN 1, 640, 200, 4, 2          [RETURN]
```

Yikes! Your monitor is suddenly blank. You see a white drag bar running left to right along the top, a pair of front-to-back gadgets in the top right corner, and the mouse pointer. The rest is empty space. It's a new blank screen. O.K., now what?

Move the mouse pointer to the drag bar, press and hold the left mouse button, then pull the mouse towards you. The new screen drags down on your monitor, exposing the Workbench screen behind. AmigaBASIC's OUTPUT window is there, exactly as you left it. Now, drag the new screen back up to the top and then left-select the black front-to-back gadget (the little black box). Zap! The Workbench screen fills the monitor and the new screen is now in the background.

Now, let's assume that you have not resized AmigaBASIC's OUTPUT window since you first fired it up. When AmigaBASIC opens, the OUTPUT window is full-size, covering the entire screen. The front-to-back gadget you see in the upper right-hand corner does not belong to a screen; it belongs to AmigaBASIC's OUTPUT window. If you select it, you will flip not screens, but windows.

So, how do you switch back to your new screen? You can resize AmigaBASIC's OUTPUT window, thus exposing the Workbench screen's front-to-back gadget and drag bar, or you can use the keyboard. Press [LEFT_AMIGA]-M and the screens flip. There's your new screen again. Now press [LEFT_AMIGA]-N. The Workbench screen flips back to the front. You will use

these keystrokes often when you want to return to the Workbench screen while a fractal is being calculated.

Left-select anywhere in AmigaBASIC's OUTPUT window (in the Workbench screen) to make sure it's active, and then enter the following command:

```
WINDOW 2, "My Window", (50,25)-(550,175), 31, 1          [RETURN]
```

Again you see the new screen, but this time you see that it also has a display area (window) surrounded by a white border and bearing the title "My Window." You should now be able to print in 16 colors. Let's see. Flip back to the Workbench screen, left-select in AmigaBASIC's OUTPUT window (in the Workbench screen) to make sure that it is active, and enter the following command:

```
FOR I=1 TO 1000: COLOR I MOD 16: PRINT I: NEXT I
[RETURN]
```

Now flip screens again. There you are—16 colors! Isn't that amazing? Your computer is counting to 1000 in 16 colors on your new screen, yet you can still flip windows and do other things on your Workbench screen. That's multitasking.

Now, flip back to the Workbench screen, left-select in AmigaBASIC's OUTPUT window (in the Workbench screen) to make sure it is activated, and enter

```
[CTRL]-C
```

Flip screens and see that the counting has stopped. Now go back to AmigaBASIC's OUTPUT window and enter

```
SCREEN CLOSE 1          [RETURN]
```

Your 16-color screen is gone. If you try to flip screens, nothing will happen.

This month's example: The SCREEN command

```
SCREEN 1, 640, 200, 4, 2
```

The first number, 1, is the screen number. Memory permitting, you can open 4 screens in AmigaBASIC, which are numbered 1 through 4. The numbers 640 and 200 obviously specify the screen resolution. The next number, 4, is the color specification. It indicates that you want 16 colors, the maximum available in 640 x 200 pixel resolution. The last number, 2, is called the "mode"; it must change depending on the resolution. In these articles I will always use a resolution of 640 x 200 pixels, meaning that the mode will always have a value of 2.

The WINDOW command

```
WINDOW 2, , (0,0)-(631,186), 0, 1
```

The first number, 2, is the window number. AmigaBASIC uses window number 1 for its own input and output, thus allowing you to enter commands like FILES, LIST, LOAD, SAVE, etc. in immediate mode. You may use window number 1 in your own programs, but it is not recommended.

The empty space between the first two commas means that the window will have no title. The numbers in parentheses are the coordinates of the upper left-hand and lower right-hand corners of the window. (More about these in a minute.)

The next number, 0, is a window attribute number. This number allows you to give your window fancy features like drag bars and sizing gadgets, but you don't need any of these right now. They take up room on the screen, and you want the maximum possible area for drawing your fractals. Let's leave the attribute number at 0. The last number, 1, is the number of the screen you want the window to appear in. It's the same number that was specified earlier in the SCREEN command.

Confusing news

Although you opened a screen with a resolution of 640 x 200 pixels, the actual size of your drawing area is 632 x 196 pixels. Now wait one minute! Since we are asking for a window that has no drag bar or sizing gadget, why don't we get the full 640 x 200 pixel area? Well, AmigaBASIC always draws a border around its windows, and that border takes up a few pixels. In my program I would prefer to have no border, but there is no easy way to get rid of it in AmigaBASIC. So, I just work with what's available.

But wait again, you say. Why isn't the drawing area 632 x 187 pixels, as specified by the numbers (631,186) in the WINDOW command? Good question. You see, when the Amiga executes a WINDOW command, it counts vertical pixels as if you wanted a drag bar, restricting the maximum number of vertical pixels to 187. There simply isn't room on the screen for a drag bar and more than 187 vertical pixels. If you ask for more, you'll get the message "illegal function call." The WINDOW command sequence in the last section opens the largest window possible. However, because you are asking for a window with no drag bar (attribute number 0), you will get the extra nine pixels that would have been used to make one. These extra pixels are available for drawing graphics. The Amiga coordinates (pixel numbers) for the lower right-hand corner of the drawing surface thus become (631,195). Confusing? You bet!

Last modification

Remember those two scaling lines at the beginning of the programs in the last two articles? They were written for a drawing surface of 618 x 187 pixels. Now they will have to be rewritten for a drawing surface of 632 x 196 pixels. Below I list everything needed in AmigaBASIC to set up and scale your 16-color window. (Note that the values I use for xmin through ymax are from this month's example.)

```
DEF FNx(x) = INT(((x-xmin)+dx/2)/dx)
DEF FNy(y) = 195 - INT(((y-ymin)+dy/2)/dy)
xmin = -0.9505
xmax = -0.8825
ymin =  0.235
ymax =  0.29
dx = (xmax-xmin)/631
dy = (ymax-ymin)/195
SCREEN 1, 640, 200, 4, 2
WINDOW 2, , (0,0)-(631,186), 0, 1
```

The new screen runs simultaneously with the Workbench screen. You can flip between them using the [LEFT_AMIGA]-N and [LEFT_AMIGA]-M keystrokes. See Listing One for the complete example that draws a 16-color fractal.

This overview of the SCREEN and WINDOW commands is rather brief, but it should be sufficient for anyone interested in running examples in future articles in this series. If you feel the need to know more, I recommend *Advanced AmigaBASIC* by Halfhill and Brannon (Compute, 1986).

True BASIC

As usual, True BASIC has a slightly easier way of doing the same thing:

```
set mode "HIGH16"
let xmin = -0.95
let xmax = -0.85
let ymin =  0.2325
let ymax =  0.3
dx = (xmax-xmin)/639
dy = (ymax-ymin)/199
set window xmin, xmax, ymin, ymax
```

The command SET MODE "HIGH16" opens both a new screen and a full-size borderless window. You do not have to open a screen first and then a window, as you must do in AmigaBASIC. In fact, you don't even have to know the difference between a screen and a window. You also get the full 640 x 200 pixel area to draw on. No modifications of any lines from previous examples are necessary.

Like AmigaBASIC, True BASIC allows you to flip between screens with the [LEFT_AMIGA]-M and [LEFT_AMIGA]-N keystrokes.

Remember, unlike AmigaBASIC's WINDOW command, TRUE BASIC's SET WINDOW command does not open a new window. Instead, it scales (directly in Cartesian coordinates!) a window that has been previously opened by the SET MODE command. Listing Three gives the complete True BASIC version of this month's example.

Specifying colors

There are 16 color registers, each of which can hold a different color chosen from a palette of 4096. In AmigaBASIC, you use the PALETTE command:

```
PALETTE 1, 15/16, 7/16, 0/16
```

This line loads pink into color register 1. The fractions represent different intensities of the basic colors red, green, and blue. All colors on the Amiga are formed from these intensity numbers. They are combined together in the register to form a single color. Intensity numbers can have 16 different values: 0/16, 1/16, 2/16, ... , 14/16, and 15/16. The fraction 0/16 represents minimum intensity, while 15/16 represents the maximum. Since there are 3 basic colors—red, green, and blue—in each register, and each of these has 16 possible intensities for each basic color, your resulting color will be one out of a possible 4096 (16 x 16 x 16 = 4096). Note that you can write these intensity numbers as decimals (e.g., 0.25 or 0.3125), but I prefer to use fractional notation.

Knowing exactly which intensity number combination will produce a certain color requires a little practice. An easy way to get started is to go to the Preferences screen on the Workbench and play with the color controls you will find there. You will see the colors change as you move each control, and its position will give you a crude idea of the color register numbers required to produce that color.

A more exact method is to use *DeluxePaint*. The [RIGHT_AMIGA]-P (Palette) command provides color controls that also indicate the exact numbers required in the numerators of the color fractions. If you don't have *DeluxePaint*, you can write a program in AmigaBASIC. There's an example on page 49 of *Advanced AmigaBASIC*, although it gives you color numbers as decimals rather than as fractions.

To specify the basic color of each register, I use the subroutine Choose.Color.Numbers:, in which I store the numerator of each color-intensity fraction. These numerators are divided by 16 in the Use.Color.Numbers subroutine, which is called later.

```
Choose.Color.Numbers:
Reg.0.Red    =   4
Reg.0.Blue   =   0
Reg.0.Green  =   4
Reg.1.Red    =   4
Reg.1.Blue   =   7
Reg.1.Green  =   0
     .
     .
Reg.15.Blue  =   0
```

You might think I'm being too pedantic by using a separate variable name for the intensity number of each basic color in each register, but I have a good reason. In future articles I will be modifying these numbers to create different artistic effects. It will be easy then to look down the list of assignments and find the register and color I want to modify.

Making modifications to a program after it is written is called program maintenance. Designing programs that are easy to maintain is a smart thing to do. When you come back to a program some time in the future, you will almost certainly have forgotten exactly how it works. Naming variables descriptively helps remind you of their purposes. I know what you're thinking: But don't descriptive variable names take up a lot of valuable memory?

How much memory is "a lot"?

Computers today have lots of memory. A 512K Amiga has plenty of room for the sort of thing I'm doing here. In general, if you can think of anything that will make your program easier to understand and maintain—it's worth it! That's what big machines are for. By using AmigaBASIC's cut and paste features, you can design easy-to-maintain programs without an inordinate amount of time on the keyboard. Type the following three lines:

```
Reg.0.Red   =
Reg.0.Green =
Reg.0.Blue  =
```

Now, left-select and drag the mouse pointer across these three lines; they will turn orange. Press [RIGHT_AMIGA]-C to copy them to the buffer (or clipboard, as some people like to call it). Place the cursor at the next empty line and press [RIGHT_AMIGA]-P. Zap! Three new lines are instantly added to your program. Press [RIGHT_AMIGA]-P 14 times. Now, using the mouse to position the cursor, type the required intensity numbers at the end of each line. Done!

Structured programming

Long programs should be organized into clearly defined parts that divide the total programming task into many simpler ones. Each part can be identified by an informative name that shows its relationship to the total programming task. To make its internal operation easier to understand, each part should contain only the code particular to its own task.

Both AmigaBASIC and True BASIC have two parts designed for this purpose— subroutines and subprograms (called functions in True BASIC). Both have their advantages and disadvantages, depending on the programming situation. I

will use subroutines mainly because they are probably already familiar to those of you who have used older BASICs on other (more primitive) computers.

So what's a subroutine?

Let me give you just a quick overview of subroutines, as more on them will follow in future articles. In short, a subroutine is like a little program within a program which is executed by the GOSUB statement. For example, this month's example contains the line:

```
GOSUB Choose.Color.Numbers
```

This causes the computer to jump to the program section labeled Choose.Color.Numbers: and to execute the program lines it finds there. The last statement of the subroutine is the RETURN statement, which tells the computer to go back and do whatever it was doing before the subroutine was called.

With subroutines, the program operation is easier to follow and understand. When you see GOSUB Choose.Color.Numbers, you recognize its purpose immediately, just by looking at its name. You don't have to go through all those color-choosing lines to see what the program is doing. Yet, if you do want to see the color-choosing lines—perhaps to modify them—you know where they are. That's structured programming.

Saving pictures to disk

In the first article of this series, I recommended you use software called *GRABBiT* to save your fractals to disk, and *DeluxePaint* to view them once they are saved. While this is certainly an easy solution that allows you to get on with the business of drawing fractals, it may not be acceptable to everyone. First of all, that software costs money! Secondly, there is always that old do-it-yourself spirit. I therefore devote the remainder of this article to explaining methods that can be used to save your fractal pictures to floppy disk in AmigaBASIC.

Before getting started, let me say that there are many ways to save data to disk, and just as many opinions among programmers as to which is the best. One person may like IFF format. Others may want a method that uses only simple, easy-to-understand AmigaBASIC commands (no complicated system or library calls). Others may already be using *GRABBiT*, and would prefer that I devote less space to this than I already have. Well, I obviously cannot satisfy all these different preferences in a single article.

IFF format in AmigaBASIC

The folks who shipped you your Amiga gave you three programs on the Extras disk with which to save graphics patterns to disk in IFF format. Unfortunately, these programs are quite complicated—certainly beyond the average user's capability, and far above the technical level of this article. Yet, guided by exact instructions, I'm sure you can get them working.

The Extras disk

Make a copy of it! Look in the BasicDemos drawer and notice three files with AmigaBASIC data icons: exec.bmap, graphics.bmap, and dos.bmap. Some early Extras disks don't have all three files. If you are missing any of them, they can be created by using the ConvertFD program, which is also in the BasicDemos drawer. I described how to do this in my "Computer Aided Instruction" article (AC V 3.9, page 74). (Incidentally, I just received my new 1.3 AmigaDOS and it came with all three files already on it.) Once you have them all, copy the files to the root directory of a blank floppy disk. If you have a single-drive system, drag their icons to the RAM: disk, replace the Extras disk with a new blank disk, then drag the icons to the blank disk. You'll do a lot less disk-swapping this way.

In the BasicDemos drawer of the Extras disk, notice the following three files, each of which has an AmigaBASIC program icon: SaveILBM, LoadACBM, and LoadILBM-SaveACBM. Copy these files to the root directory of the same disk you used above. Finally, if you haven't done so already, copy to the same disk—as AmigaBASIC itself—the square box in the root directory of the Extras disk. You will be using one of these programs by tacking it on to the end of this month's example Listing One (using AmigaBASIC's MERGE command). However, before doing so you must make sure that the SaveILBM program is saved in ASCII format, as required by the MERGE command. Here's how to do that:

```
1.) Fire up AmigaBASIC from the above disk  (double-click its
    icon)
2.) In the OUTPUT window type:  LOAD "SaveILBM"
3.) In the OUTPUT window type:  SAVE "SaveILBM",A
4.) Quit AmigaBASIC
```

You can now consider this disk a master AmigaBASIC work disk. You can save all your work to a copy of this disk; there's plenty of room. Make new copies of it when you want to start other projects. Now enter this month's example Listing One. Notice the complicated-looking lines in the Save.to.Disk subroutine. Enter this stuff carefully! Now, with the program still in memory, activate AmigaBASIC's OUTPUT window and enter

```
MERGE "SaveILBM"
```

Presto! The SaveILBM program has been tacked on to the end of Listing One. Save it and you're done!

If you look at this newly merged code, you'll probably agree that it's quite complicated. I might also mention that not all of it will be used. But don't concern yourself with that now; the parts that are not used will not hurt anything.

Testing the program

This month's example fractal takes a very long time to generate, so you'll want to make sure the Save.to.Disk routine works before committing your Amiga to the job. Do a trial run first. Fire up the program (double-click the program icon). You will immediately be asked to enter the name under which the fractal will be saved. The program will then start to generate your fractal very slowly. You can quit at any time by pressing the [F-10] function key. Press it now, just to see if it works.

Successful test

The disk drive will make some crunching noises while saving the screen to disk. From the Workbench, you will not be able to see the file containing the picture. No icon was generated for it. But you can check it by using either *DeluxePaint*, or the LoadILBM-SaveACBM program.

DeluxePaint

Put *DeluxePaint* in df0: and your work disk in df1:. Fire up *DeluxePaint* as if you wanted to look at a 16-color medium-

AMZ 10-89

Order Toll Free 800-558-0003

ComputAbility Consumer Electronics

Since 1982

Order Toll Free 800-558-0003

IVS Trumpcard Hard Drive Packages for A2000 Series

Seagate

ST-157N 49 MEG	519 DEL
ST-177N 60 MEG	749 DEL
ST-277N-1 60 MEG	819 DEL
ST-296N 80 MEG	649 DEL

Quantum Pro Drive

40 MEG SCSI Pro Drive	629 DEL
80 MEG SCSI Pro Drive	949 DEL
100 MEG SCSI Pro Drive	1029 DEL

these kits include IVS Trumpcard 8C8I hard drive ctrl, cable, software and **FREE delivery** in the contiguous USA This is not anassembly kit ! It is a package.

Hard Drive Cards (A-2000)

Seagate

ST-157N 49 MEG	539 DEL
ST-177N 60 MEG	769 DEL

Quantum Pro Drive

40 MEG SCSI Pro Drive	649 DEL
80 MEG SCSI Pro Drive	969 DEL
100 MEG SCSI Pro Drive	1049 DEL

these cards include IVS Trumpcard ctrlr, mounting brkt, cbl, software, and **FREE delivery** in the contiguous USA.

IVS INFINIT 40

Removable Media Cartridge Drive Packages

INFINIT40/I	1199
INFINIT40S *	1299
INFINIT40D *	1399

* Requires Trumpcard or Trumpcard 500

Trumpcard 500 Hard Drive Packages for Amiga 500

Seagate

ST-157N-49 Meg	579 Delivered!
ST-177N-60 Meg	799 Delivered!

Quantum Pro Drive

40 Meg SCSI Pro Drive -	679 DELIVERED
80 Meg SCSI Pro Drive -	999 DELIVERED
100 Meg SCSI Pro Drive -	1079 DELIVERED

These Packages include Trumpcard 500 enclosure, IVS Trumpcard ,SCSI hd drive Ctrl, Software & free delivery in the contiguous USA.

Trumpcard 500 Ram Expander available 512k • 1 MEG • 2 MEG • 4 MEG

Seagate SUPER SALE

ST-125N 20 Meg - SCSI	279
ST-157N 49 Meg - SCSI	349
ST-177N 60 Meg - SCSI	569
ST-225N 20 Meg - SCSI	265
ST-251N 40 Meg - SCSI	375
ST-277N-1 60 Meg - SCSI	449
ST-296N 80 Meg - SCSI	479

Quantum

Pro Drives

40 Meg SCSI Pro Drive	529
80 Meg SCSI Pro Drive	899
100 Meg SCSI Pro Drive	929

XETEC Fast Card

Hard Drive Packages for Amiga 2000

Seagate

ST-157N-49 Meg	519 Del
ST-177N-60 Meg	749 Del

Quantum Pro Drive

40 Meg SCSI Pro Drive	629 DEL
80 Meg SCSI Pro Drive	949 DEL
100 Meg SCSI Pro Drive	1029 DEL

Card features Xetec's DMAx hard, autoboot ROM,auto-config circuitry..25 pin SCSI conn., disk, utilities manual.

FREE DELIVERY IN THE CONTIGUOUS 48 STATES

XETEC *Fast Card System*

Hard Drive Packages for Amiga 500

Seagate

ST-157N-49 Meg	699 Del
ST-177N-60 Meg	919 Del

Quantum Pro Drive

40 Meg SCSI Pro Drive	819 DEL
80 Meg SCSI Pro Drive	1139 DEL
100 Meg SCSI Pro Drive	1219 DEL

System includes Adaptor with autoboot ROM, enclosure, manual, Fast Trak disk with utilities & shielded SCSI cable. With power supply and fan

FREE DELIVERY IN THE CONTIGUOUS 48 STATES

CALL US FOR ALL YOUR COMMODORE AMIGA™ NEEDS

MASTER 3A Disk Drive

1010 Compatible **$129**

*Free Delivery to the Contiguous States

VIDEO PACKAGE

PANASONIC 1410 CAMERA
16MM LENS WITH VARIABLE IRIS
COPYSTAND WITH LIGHTS
DIGIVIEW GOLD
$419 DELIVERED!

PRINTERS

Panasonic 1180 NEW NEW NEW	185
Panasonic 1191 NEW NEW NEW	CALL
Panasonic 1124 NEW 24 Pin	319
Star NX-1000	159
Star NX-1000 Multi Font 2 NEW NEW NEW	179
Star NX-1000 Rainbow (color)	219
Star NX-2400	275

Modems

Amiga 1680 Modem	99
Avatex 1200E	69
Avatex 2400	139
Avatex 2400E	109
Avatex 9600 External NEW!	839
Supra 2400zi Internal (A2000)	139

Supra 2400 Ext. Modem & Cable

•$129•

FREE DELIVERY
to the 48 Contiguous States

AMIGA SOFTWARE

1943	25.95	Combat Course	25.95	Fiendish Freddy	31.95	Jinks	18.95
3D Options	31.95	Comic Setter	59.95	Fine Print	28.95	Joan of Arc	28.95
3D Pool	22.95	Comic Art Disks (each)	20.95	Fire Zone	22.95	Joe Blade	19.95
A/C Basic	128.95	Cantra	25.95	Flight Simulator II	31.95	Journey	31.95
A-Talk 3	64.95	Cribbage King/Gin King	38.95	European Scenery Disk	19.95	Jug	24.95
Action Fighter	24.95	Crystal Clear Turbo	17.95	Flow	59.95	Kampfgruppe	38.95
Advantage, The	46.95	Curse Buster	19.95	Focus Video	107.95	Karafonts Headlines 1 or 2	44.95
Adventures of Sinbad	28.95	Daily Double Horse Racing	19.95	Forgotten Worlds	24.95	Karafonts Subhead	44.95
After Burner	31.95	Dark Side	24.95	Form In Flight	69.95	Keef the Thief	31.95
Alien Syndrome	24.95	Data Retrieve Pro	179.95	Formation	44.95	Kind Words V2.0	67.95
AmigaDOS TooBox	38.95	Data Storm	24.95	Fright Night	24.95	King Arthur's Quest/Excal.	31.95
Animation Sound Track	69.95	Deathbringer	24.95	Front Page	74.95	King's Quest IV	31.95
Animation Titler	48.95	Defender Of The Crown	31.95	Future Sound 500	89.95	Kingdoms of England	26.95
Anamagic	58.95	Deja Vu: 1 or 2	31.95	Galactic Conqueror	28.95	Knights of Legend	31.95
Android Decision	23.95	Deluxe Music Constr 2.0	61.95	Games : Winter Ed	13.95	Kristal, The	31.95
Anim 4-D Jr.	67.95	Deluxe Paint III	97.95	Gauntlet II	31.95	Last Duel	24.95
Anim. Editor	39.95	Deluxe PhotoLab	97.95	Gettysburg	38.95	Last Inca, The	24.95
Animation Multiplane	82.95	Deluxe Print II	51.95	GFA Basic 3.0	87.95	Leathernecks	24.95
Animation Rotoscope	48.95	Deluxe Productions	128.95	Gold of the Realm	24.95	LED Storm	24.95
Animator Apprentice	174.95	Deluxe Video III	95.95	Golds of the Americas	26.95	Leisure Suit Larry	25.95
Annals of Rome	23.95	Deluxe Video V1.2	84.95	Gold Runner 2	24.95	Leisure Suit Larry II	31.95
Archipelagos	24.95	Demon's Winter	26.95	Gold Rush	24.95	Leonardo	25.95
Arexx	28.95	Denaris	24.95	Grabbit	18.95	License to Kill	21.95
Arkanoids	17.95	Design 3D	59.95	Grand Prix Circuit	27.95	Life and Death	32.95
Aunt Artic Advent.	25.95	Designasaurus	30.95	Gunship	35.95	Lights Camera Action	46.95
Artura	25.95	Devon Aire Diamond Caper	31.95	Handball II	28.95	Logiworks	184.95
Attack on London	22.95	Digidroid	59.95	Harley Davidson	31.95	Lords of the Rising Sun	30.95
Award Maker	29.95	Digipaint3.0	64.95	Harpoon	25.95	Magellan 1.1	114.95
Awesome Arcade Pack	32.95	Digiview Gold	124.95	Harrier Combat Simulator	31.95	Magic Johnson 512	24.95
Axe of Rage	31.95	Digiwork 3D	82.95	Heat Wave	28.95	Magic Johnson 1 Meg	31.95
B.A.D.	29.95	Dino Wars	CALL	Heavy Metal	28.95	Manhunter - New York	31.95
Baal	19.95	Dinosaur Discovery Kit	25.95	Heros of the Lance	25.95	Maniac Mansion	29.95
Balance of Power 1990	31.95	Disk 2 Disk	28.95	Hillsfar	31.95	Marble Madness	13.95
Bar Games	28.95	Disk Master	28.95	Hole In One	21.95	Mavis Beacon Typing	31.95
Bard's Tale	31.95	Disk Mechanic	54.95	Hole In One - Course Disk	13.95	Mean 18	27.95
Hint Disk	15.95	Dive Bomber	14.95	Hostage	29.95	MediaLine Clip Art	21.95
Bards Tale II	38.95	Doc 2 Doc	31.95	Hunt For Red October	31.95	Menace	19.95
Batman	28.95	Double Dragon	25.95	Hybris	22.95	Miami Vice	24.95
Battle Chess	29.95	Doug's Math Aquarium	46.95	Hyper Dome	22.95	Michtron Hit Disk #1	32.95
Battle Hawks 1942	31.95	Dragon Force	31.95	I Ludicrous	22.95	Micro Fiche Filer Plus	114.95
BattletechThe Cr Hk Inc.	31.95	Dragon's Lair	33.95	Impossible Mission 2	14.95	Mid Magic	CALL
Baud Bandit	30.95	Dragon's Lair II	CALL	Indiana Jones-Action Game	26.95	Mid Rec Studio V1.1	39.95
Berzerk 3	22.95	Draw 2000	159.95	Indiana Jones-Graphic Adv.	31.95	Mid View	23.95
Beyond Dark Castle	21.95	Dungeon Master	24.95	Indoor Sports	30.95	Might and Magic II	38.95
Black Jack Academy	24.95	Dungeon Quest	31.95	Innovatools 2	48.95	Mighty Nerds	31.95
Blastaballs	CALL	Dunlap Utilities	47.95	Instant Music	13.95	Mindroll	19.95
Blood Money	25.95	Dynamic Drums	46.95	Inter Font	69.95	Mixed Up Mother Goose	19.95
Blue Angels	28.95	Earl Weaver Baseball	31.95	Interchange	66.95	Modeler 3D	66.95
Boomerald	25.95	Commissioners Disk	14.95	IntroCad	46.95	Moebius	38.95
Bride of the Robot	24.95	MLBPA Stats	14.95	Investors Advantage	59.95	Movie Setter	59.95
Broadcast Titler	184.95	Elan Performer	35.95	Invision	134.95	Music Studio 2.0	49.95
Bubble Bobble	22.95	Empire	31.95	Ishido	31.95	Music X	169.95
Calamus	274.95	Excellence 1 Meg	158.95	It Came From The Desert	31.95	My Paint Data Disk	17.95
California Games	13.95	Exolon	22.9	Jack Nicklaus Golf	30.95	Netherworld	22.95
Calligrapher	72.95	Eye of Horus	24.95	JN Courses of 1989	14.95	Neuromancer	28.95
Can Do	84.95	F-19 Stealth Fighter	CALL	JN Course Disc II	14.95	New York Warriors	CALL
Captain Blood	31.95	F-40 Pursuit Sim.	28.95	Jet	31.95	Night Force	28.95
Captain Fizz	19.95	FA/18 Interceptor	31.95				
Carmen SanDiego/ Time	28.95	Facc II	20.95				
Carmen SanDiego/ USA	31.95	Faery Tale Adven	29.95				
Centerfold Squares	18.95	Falcon	29.95				
Chessmaster 2000	28.95	Operation Counterstrike	16.95				
Chessmaster 2100	31.95	Fantavision	37.95				
City Desk 2.0	112.95	Fast Break	27.95				
Clue Master Detective	25.95	Federation	31.95				
Colony	31.95	Ferrari Formula 1	31.95				

Novelty Fonts	48.95	Qix	22.95	Sword of Sodan	28.95
Nuclear War	32.95	Quarter Back	42.95	Swords of Twilight	32.95
Obliterator	26.95	Rambo III	22.95	T.V. Sports Basketball	30.95
Off Shore Warrior	24.95	Rampage	24.95	T.V. Sports Football	30.95
Omni Basketball	31.95	Rastan	25.95	Talespin	31.95
Omniplay Baseball: 5 on 5	31.95	Raw Copy	34.95	Tanglewood	24.95
On Line Platinum	58.95	Realm of the Warlock	21.95	Technocop	31.95
Operation Wolf	22.95	Red Lightning	38.95	Temple of Doom	31.95
Optics ..... CALL FOR SALE PRICE		Reel Fish'n	29.95	Tetraquest	24.95
Oswald	23.95	Renegade	22.95	Tetris	20.95
Outrun	31.95	Return To Atlantis	13.95	Text Ed Plus	48.95
Page Renderer 3-D	94.95	Revenge of Defender	21.95	Text Drive 2 - The Duel	27.95
Page Setter	59.95	Rhyming Notebook	31.95	California Chall.	14.95
Page Stream	127.95	Rick Dangerous	31.95	European Chall.	14.95
Dot Matrix Font1-5 ..... ea.	24.95	Rick Da vis Soccer 1Meg	29.95	Supercars	14.95
Page Thinker	59.95	Rick Davis Soccer 512K	24.95	Thexder	22.95
Pagefipper Plus/FX	87.95	Ring Side	24.95	Their Finest HourBattle Britain	38.95
Pen Pal	84.95	Ringling Bro's Circus	24.95	Thinker	57.95
Phasar	54.95	Road Challenge	24.95	Third Courier	28.95
Photon Paint 2.0	87.95	Robocop	28.95	Three Demon	69.95
Photon Paint Surface	19.95	Rocket Ranger	31.95	Thunder Blade	29.95
Photon Video : Cel Anim.	87.95	Romantic Encl @ Dome	25.95	Times of Lore	24.95
Photon Video Transport	169.95	Rubicon Alliance	20.95	Titan	28.95
Pinball Wizard	22.95	RVF Honda	24.95	Tom & Jerry	CALL
Pioneer Plague	26.95	Savage	22.95	Tower Toppler	31.95
Pixel Script	92.95	Scavenger	CALL	Transcripts	41.95
Planet of Lust	24.95	Scrabble	26.95	Trials of Honor	32.95
Platoon	27.95	Scrabble Platinum	87.95	Turbo	15.95
Pocket Rockets	25.95	Securities Analyst	49.95	Turbo Silver 3.0 1 Meg	112.95
Police Quest 1 OR 2	31.95	Sex Vixens-Outer Space	22.95	TV Show	64.95
Pool of Radiance	31.95	Shadowgate	31.95	TV Text	64.95
Populous	32.95	Shinobi	31.95	Twilight Zone	24.95
Ports of Call	28.95	Shogun	31.95	Twilight's Ransom	19.95
Postscript Fonts A,B or C . ea	24.95	Shuffleputk Cafe	24.95	Ultima 4	38.95
Power Windows 2.5	54.95	Side Arms	24.95	Ultima II	24.95
Powerdrome	26.95	Silent Service	24.95	Undersea Comando	31.95
Precisely	48.95	Sim City	28.95	Universe 3	31.95
Print Master Plus	24.95	Skate Wars	31.95	Vampire's Empire	28.95
Pro Script	28.95	Sky Blaster	18.95	Video Effects 3-D	113.95
Pro Soccer	22.95	Sky Chase	24.95	Video Fonts	28.95
Pro Sound Design	96.95	Sky Fox II	13.95	Video Page Titler	86.95
Pro Video Gold	174.95	Sky Shark	22.95	Video Titler	84.95
Pro Write 2.5	69.95	Slipped Stream	17.95	VideoScape 3D 2.0	115.95
Pro-Draw	112.95	Softwood File II	62.95	Interior Design	22.95
Professional Page	228.95	Sorceror Lord	23.95	Vindicators	CALL
Structured Clip Art	36.95	Space Harrier	31.95	Virus Infection Protection	28.95
Templates	36.95	Space Quest 1 or 2	31.95	Vista 3-D	87.95
Profonts 1 & 2	21.95	Space Quest 3	CALL	VTX Online	51.95
Project D+ Backup	30.95	Space Racer	19.95	W Shell	28.95
Project Master	129.95	Speedball	24.95	War In Middle Earth	29.95
Project Neptune	21.95	Spinworld	18.95	Wayne Gretzky Hockey	29.95
Prospector Mazes of Zor	25.95	Star Command	31.95	Weird Dreams	24.95
Pub Games	25.95	Star Ray	29.95	Who, What, When	64.95
Publisher's Choice	114.95	Star Trek: The Final Front.	36.95	Wind Walker	25.95
Publisher Plus	64.95	Star Wars	24.95	Wings Of Fury	25.95
Purple Saturn	19.95	Stellar Crusade	36.95	Works Platinum, The	CALL
Puzzle Storybook	25.95	Street Fighter	25.95	World Atlas	35.95
		Strip Poker II	23.95	World Tour Golf	13.95
		Studio Magic	59.95	X-Specs	84.95
		Sub Battle Sim	14.95	Zak McKracken	29.95
		Super Star Basketball	CALL	Zany Golf	26.95
		Superbase Personal	46.95	Zoetrope	82.95
		Superbase Personal 2	87.95	Zork Zero	38.95
		SuperBase Pro 3.0	189.95	Zynaps	22.95
		Superplan	87.95		
		Superstar Ice Hockey	31.95		

MISCELLANEOUS HARDWARE

AMI GEN	$135	CHINON FB 354 INT. DR.	$99
SUPER GEN	$669	COLOR SPLITTER	$89
SUPER GEN 2000S INT GENLOCK .	$1449	GOLD DISK SCANNER	$899
MAGNI 4004 GEN LOCK	$1389	IMG SCAN	$105
FLICKER FIXER	$455	SCAN LOCK	$819
FLICKER FIXER GENLOCK OPT.	$39	GVP 68030 ACC. BD	CALL
PERFECT SOUND	$69		

VISA MasterCard

NO CREDIT CARD SURCHARGE

WI ORDERS AND INFORMATION
414-357-8181 FAX 414-357-7814
P.O. BOX 17882 Milwaukee, WI 53217

ORDERING INFORMATION: Specify system. For fast delivery send cashier's check or money order. Personal and company checks allow 14 business days to clear. School P.O.'s welcome. C.O.D. charges are $4.00. In Continental U.S.A. include $3.00 for software orders 5% shipping for hardware, minimum $6.00. MasterCard and Visa orders please include card #, expiration date and signature. WI residents please include 5% sales tax. HI, AK, FPO, APO,Puerto Rico and Canadian orders, please add 8% shipping, minimum $6.00. All other foreign orders add 15% shipping, minimum $15.00. All orders shipped outside the Continental U.S.A. are shipped first class insured U.S. mail. If foreign shipping charges exceed the minimum amount, you will be charged the additional amount All goods are new and include factory warranty. We do not guarantee compatability & version #s. Due to our low prices all sales are final. All Defective returns must have a return authorization number. Please call (414) 357-8181 to obtain an R.A. # or your return will not be accepted. Prices and availability subject to change without notice. Shipping and handling are non-refundable. We ship the latest versions available to us, updates must be handled by end user directly with the manufacturer.

resolution picture. Next, select Load from the pulldown menu, choose df1:, select whatever filename you saved your picture under, and then select Load. You will see the messages "Your fractal is being generated," and "Press [F-10] to quit." That's the screen you just saved to disk. It works!

LoadILBM-SaveACBM

Fire up the LoadILBM-SaveACBM program (double-click its icon). At the IFF ILBM filespec prompt, enter the name you gave your fractal. At the ACBM filespec prompt just press [RETURN], entering nothing. The disk will whir and you'll see your picture for about 15 seconds before it disappears. If you want to view the picture for a longer period of time, you have to modify the LoadILBM-SaveACBM program. Here's what to do:

```
1.) Fire up AmigaBASIC by clicking on its icon.
2.) In the OUTPUT window, type:  LOAD "LoadILBM-SaveACBM"
3.) In the OUTPUT window, type:  LIST Mcleanup.  The LIST
    window appears and the part of the program that you
    have to modify is on the screen.
4.) Delete the second line, which reads:  FOR de = 1 TO
    20000:NEXT
    Replace it with:  WHILE INKEY$ <> CHR$(138):WEND
    This will cause the picture to remain on the screen until
    you press the [F-10]  function key.
5.) Save the program.
```

Using GRABBiT

When the program finishes calculating your fractal, it will remain on the screen so you can save it to disk with the [CTRL]-[ALT]-S command (*GRABBiT*'s save command). When you are done, press the [F-10] function key. You will be returned to the Workbench.

Other ways to save pictures

As I said earlier, there are many ways to save pictures. The one presented here has the obvious advantage of letting you look at your pictures using *DeluxePaint*. In future articles, you will be modifying the colors of your fractals to produce your own artistic effects, and *DeluxePaint* will prove to be an excellent tool for doing that.

But you still might have an empty feeling about all of this because you really didn't write the program yourself, but copied it from the Extras disk. I feel the same way: I prefer to program the computer myself. You can always save your pictures using AmigaBASIC'S GET and PUT commands. The GET command can read graphics information directly from the screen, and it is not difficult to write a program to save that data to disk.

This method, although somewhat slower in AmigaBASIC, is much easier to understand than saving in IFF format. I wrote programs that use this method in both AmigaBASIC and True BASIC. However, because of space constraints, I'll have to present these in a future article. The method is also described in *Advanced AmigaBASIC*.

Still not satisfied?

Maybe you want to do it yourself but still want to save your pictures in IFF format. You can! I would recommend the book *AmigaBASIC: Inside and Out* by Rugheimer and Spanik (Abacus, 1988). With example programs less complicated than the SaveILBM supplied by Amiga, it demonstrates how to save your files in IFF format.

Saving in True BASIC

I assume that True BASIC users are more likely to want to use *GRABBiT*, and that they are not interested in seeing a lot of space devoted to this topic. Again, if there is sufficient reader interest, I can certainly do so in a future article.

This month's example

This month's example was modeled after a picture in the book *The Beauty of Fractals* by Peitgen and Richter (Springer-Verlag, 1986, p. 79). It gives you a chance to compare the Amiga's performance against that of a "professional computer." Naturally, pictures from the book are more impressive, as they were produced using a higher resolution screen and 256 colors. Nevertheless, the Amiga does a pretty good job. The example demonstrates that you can produce very complex fractals on your Amiga, using relatively short programs in BASIC.

Execution time for this example is considerably longer than it is for the examples of previous months:

```
Approximate Execution Times

AmigaBASIC   36 hours
True BASIC   24 hours
A/C BASIC    15 hours
```

Why does it take so long? The reason is that this month's example represents a much greater magnification of the Mandelbrot set than do any of the previous months' examples, thus requiring more processing time.

Future fractals

Next time, we will continue towards our goal of understanding how these fractals are produced by seeing exactly how to program a computer to translate algebraic equations into graphics patterns.

Listing One: Fractal-AmigaBASIC

```
REM ************************************************************
REM *                                                          *
REM *                    Sixteen Color Fractal                 *
REM *                                                          *
REM *                      by Paul Castonguay                  *
REM *                                                          *
REM ************************************************************

DEF FNx(x) = INT(((x-xmin)+dx/2)/dx)
DEF FNy(y) = 195 - INT(((y-ymin)+dy/2)/dy)
xmin = -.9505
xmax = -.8825
ymin = .235
ymax = .29
dx = (xmax-xmin)/631
dy = (ymax-ymin)/195

SCREEN 1, 640, 200, 4, 2
WINDOW 2, , (0,0)-(631,186), 0, 1

GOSUB Choose.Color.Numbers
GOSUB Use.New.Colors

Crunch = 800
M = 4
CLS

' ************************************************************
' *         Delete next two lines if you're using GRABBiT    *
' ************************************************************
LOCATE 23, 20
INPUT "Enter name to save fractal "; ILBMname$

LOCATE 10, 23
PRINT "... Fractal is being generated ..."
```

AmiEXPO

THE AMIGA

PERSONAL COMPUTER SHOW

Come See The California Goldrush!

October 20-22, 1989

Santa Clara Convention Center

Santa Clara, CA

Over 10,000 Attendees and 120 Amiga Companies Will Be There.
STRIKE IT RICH AT AmiEXPO-CALIFORNIA!

Admission includes the Exhibition, Seminars, Keynotes & Amiga Artists Theatre!
120 Amiga Exhibitors Featuring State of the Art Software and Hardware, at the lowest prices!

Master Classes Available in Amiga Graphics, Video, Programming, Animation, Music and Publishing!
Seating for Master Classes is limited; call for schedule and availability before registering.
PRE-REGISTRATION DEADLINE IS OCTOBER 6, 1989

For Hotel Reservations Call the DoubleTree Hotel at (408) 986-0700.
Hotel reservations deadline: September 20th. 1989.
For discounted airfares, call American Airlines at (800) 433-1790 and give them this ID: S-83536.

REGISTER TODAY!

Register by Mail or Call 800-32-AMIGA Nationwide (or 212-867-4663)
For Your Ticket to The Amiga Event!

Yes, I want to come to AmiEXPO - California

___ Friday ___ Saturday ___ Sunday

Registration is $5 Additional At The Door

One day - $15 _____
Two days - $20 _____
Three days - $25 _____

Master Class(es) - List Class and Time - $60 Each

_____ _____
_____ _____
_____ _____
_____ _____

Total Amount Enclosed _____

NAME _____
COMPANY _____
ADDRESS _____
_____
CITY _____ STATE _____ ZIP _____

For ___ MasterCard or ___ VISA Payment
Expiration Date _____
Account Number _____
Name as it appears on card: _____
Signature _____

Make Check or Money Order Payable to:
AmiEXPO 211 E. 43rd St., Suite 301
New York, NY 10017

Circle 115 on Reader Service card.

```
LOCATE 12, 30
PRINT "Enter [F-10] to QUIT"

FOR j=ymin TO ymax+dy/2 STEP dy
  FOR i = xmin TO xmax+dx/2 STEP dx
    GOSUB Calculate
    GOSUB Select.Color
    PSET (FNx(i),FNy(j))
    IF INKEY$ = CHR$(138) THEN GOTO Done
  NEXT i
NEXT j
WHILE INKEY$ <> CHR$(138)
WEND

Done:

' ********************************************************************
' *             Delete next line if you're using GRABBiT             *
' ********************************************************************
GOSUB Save.To.Disk

WINDOW CLOSE 2
SCREEN CLOSE 1
SYSTEM

Calculate:
  x = 0
  y = 0
  k = 0
  r = 0
  WHILE r<=M AND k<Crunch
    xk = x*x - y*y + i
    y = 2*x*y + j
    x = xk
    k = k+1
    r = x*x + y*y
  WEND
RETURN

Select.Color:
  IF k = Crunch THEN
    COLOR 15
  ELSEIF k>200 THEN
    COLOR 14
  ELSEIF k>130 THEN
    COLOR 13
  ELSEIF k>73 THEN
    COLOR 14
  ELSEIF k>50 THEN
    COLOR 12
  ELSEIF k>44 THEN
    COLOR 11
  ELSEIF k>39 THEN
    COLOR 10
  ELSEIF k>30 THEN
    COLOR 9
  ELSEIF k=30 THEN
    COLOR 8
  ELSEIF k=29 OR k=27 THEN
    COLOR 7
  ELSEIF k=28 THEN
    COLOR 6
  ELSEIF k=25 OR k=23 OR k=21 THEN
    COLOR 5
  ELSEIF k=26 OR k=24 OR k=22 THEN
    COLOR 4
  ELSEIF k=19 OR k=17 THEN
    COLOR 3
  ELSEIF k=20 THEN
    COLOR 2
  ELSEIF k MOD 2 = 0 THEN
    COLOR 1
  ELSEIF k MOD 2 = 1 THEN
    COLOR 0
  ELSE
    COLOR 0
  END IF
RETURN


Choose.Color.Numbers:
  LOCATE 10,20
  PRINT "... Please wait while I adjust colors ..."
  Reg.0.Red   =  2
  Reg.0.Green =  0
  Reg.0.Blue  =  4
  Reg.1.Red   =  5
  Reg.1.Green =  7
  Reg.1.Blue  =  3
  Reg.2.Red   =  3
  Reg.2.Green =  5
  Reg.2.Blue  =  0
  Reg.3.Red   =  4
  Reg.3.Green =  6
  Reg.3.Blue  =  7
  Reg.4.Red   =  0
  Reg.4.Green =  2
  Reg.4.Blue  =  5
  Reg.5.Red   =  6
  Reg.5.Green =  6
  Reg.5.Blue  =  9
  Reg.6.Red   =  3
  Reg.6.Green =  8
  Reg.6.Blue  =  0
  Reg.7.Red   =  7
  Reg.7.Green =  6
  Reg.7.Blue  =  9
  Reg.8.Red   =  0
  Reg.8.Green =  8
  Reg.8.Blue  =  5
  Reg.9.Red   =  0
  Reg.9.Green =  14
  Reg.9.Blue  =  14
  Reg.10.Red   = 8
  Reg.10.Green = 13
  Reg.10.Blue  = 13
  Reg.11.Red   = 13
  Reg.11.Green = 12
  Reg.11.Blue  = 14
  Reg.12.Red   = 15
  Reg.12.Green = 0
  Reg.12.Blue  = 0
  Reg.13.Red   = 15
  Reg.13.Green = 7
  Reg.13.Blue  = 0
  Reg.14.Red   = 15
  Reg.14.Green = 14
  Reg.14.Blue  = 0
  Reg.15.Red   = 0
  Reg.15.Green = 0
  Reg.15.Blue  = 0
RETURN

Use.New.Colors:
  PALETTE 0, Reg.0.Red/16, Reg.0.Green/16, Reg.0.Blue/16
  PALETTE 1, Reg.1.Red/16, Reg.1.Green/16, Reg.1.Blue/16
  PALETTE 2, Reg.2.Red/16, Reg.2.Green/16, Reg.2.Blue/16
  PALETTE 3, Reg.3.Red/16, Reg.3.Green/16, Reg.3.Blue/16
  PALETTE 4, Reg.4.Red/16, Reg.4.Green/16, Reg.4.Blue/16
  PALETTE 5, Reg.5.Red/16, Reg.5.Green/16, Reg.5.Blue/16
  PALETTE 6, Reg.6.Red/16, Reg.6.Green/16, Reg.6.Blue/16
  PALETTE 7, Reg.7.Red/16, Reg.7.Green/16, Reg.7.Blue/16
  PALETTE 8, Reg.8.Red/16, Reg.8.Green/16, Reg.8.Blue/16
  PALETTE 9, Reg.9.Red/16, Reg.9.Green/16, Reg.9.Blue/16
  PALETTE 10, Reg.10.Red/16, Reg.10.Green/16, Reg.10.Blue/16
  PALETTE 11, Reg.11.Red/16, Reg.11.Green/16, Reg.11.Blue/16
  PALETTE 12, Reg.12.Red/16, Reg.12.Green/16, Reg.12.Blue/16
  PALETTE 13, Reg.13.Red/16, Reg.13.Green/16, Reg.13.Blue/16
  PALETTE 14, Reg.14.Red/16, Reg.14.Green/16, Reg.14.Blue/16
  PALETTE 15, Reg.15.Red/16, Reg.15.Green/16, Reg.15.Blue/16
RETURN


REM ********************************************************************
REM *        delete from here to end if you are using GRABBiT          *
REM ********************************************************************

Save.To.Disk:

    ccrtDir%   = 0
    ccrtStart% = 0
    ccrtEnd%   = 0
    ccrtSecs&  = 0
    ccrtMics&  = 0

    DIM bPlane&(5), cTabSave%(32)

    DECLARE FUNCTION xOpen&  LIBRARY
    DECLARE FUNCTION xRead&  LIBRARY
    DECLARE FUNCTION xWrite& LIBRARY
    DECLARE FUNCTION AllocMem&() LIBRARY

    LIBRARY "dos.library"
    LIBRARY "exec.library"
    LIBRARY "graphics.library"

    GOSUB SaveILBM

RETURN

' ********************************************************************
' *                                                                  *
' *          Add Extras:Demos/SaveILBM to end of listing             *
' *                                                                  *
' *             using MERGE command from AmigaBASIC                 *
' *                                                                  *
' *                  OUTPUT window  (see text)                       *
' *                                                                  *
' ********************************************************************
```

Listing Two: Fractal_TB

```
! ********************************************************************
! *                                                                  *
! *                      Sixteen Color Fractal                       *
! *                                                                  *
! *                   written for AMAZING COMPUTING                  *
! *                                                                  *
! *                        by Paul Castonguay                        *
! *                                                                  *
! *                         December 2, 1988                         *
! *                                                                  *
! ********************************************************************

set mode "HIGH16"
let xmin = -.9505
let xmax = -.8825
let ymin =  .235
let ymax =  .29
let dx = (xmax-xmin)/639
let dy = (ymax-ymin)/199

call Choose_Color_Numbers
call Use_New_Colors

set window xmin, xmax, ymin, ymax

let Crunch = 800
let M = 4

set cursor 10,23
print "... Fractal is being generated ..."

for j=ymin to ymax+dy/2 step dy
    for i=xmin to xmax+dx/2 step dx
        call Calculate
        call Select_Color
        plot points: i,j
    next i
next j


! Wait for operator to press function key [F-10]
call Wait_Response


sub Calculate
        let x=0
        let y=0
        let k=0
        let xk=0
        let r=0
        do while (r<M and k<Crunch)
           let xk = x*x - y*y +i
           let y = 2*x*y + j
           let x = xk
           let k = k+1
           let r = x*x + y*y
        loop
end sub

sub Select_Color
        select case k
               case 800
                    set color 15
               case 201 to 799
                    set color 14
               case 131 to 200
                    set color 13
               case 74 to 130
                    set color 14
               case 51 to 73
                    set color 12
               case 45 to 50
                    set color 11
               case 40 to 44
                    set color 10
               case 31 to 39
                    set color 9
               case 30
                    set color 8
               case 29, 27
                    set color 7
               case 28
                    set color 6
               case 25, 23, 21
                    set color 5
               case 26, 24, 22
                    set color 4
               case 19, 17
                    set color 3
               case 20
                    set color 2
               case 18, 16, 14, 12, 10, 8, 6, 4, 2, 0
                    set color 1
               case 15, 13, 11, 9, 7, 5, 3, 1
                    set color 0
        end select
end sub

sub Choose_Color_Numbers
    let Reg_0_Red    = 2
    let Reg_0_Green  = 0
    let Reg_0_Blue   = 4
    let Reg_1_Red    = 5
    let Reg_1_Green  = 7
    let Reg_1_Blue   = 3
    let Reg_2_Red    = 3
    let Reg_2_Green  = 5
    let Reg_2_Blue   = 0
    let Reg_3_Red    = 4
    let Reg_3_Green  = 6
    let Reg_3_Blue   = 7
    let Reg_4_Red    = 0
    let Reg_4_Green  = 2
    let Reg_4_Blue   = 5
    let Reg_5_Red    = 6
    let Reg_5_Green  = 6
    let Reg_5_Blue   = 9
    let Reg_6_Red    = 3
    let Reg_6_Green  = 8
    let Reg_6_Blue   = 0
    let Reg_7_Red    = 7
    let Reg_7_Green  = 6
    let Reg_7_Blue   = 9
    let Reg_8_Red    = 0
    let Reg_8_Green  = 8
    let Reg_8_Blue   = 5
    let Reg_9_Red    = 0
    let Reg_9_Green  = 14
    let Reg_9_Blue   = 14
    let Reg_10_Red   = 8
    let Reg_10_Green = 13
    let Reg_10_Blue  = 13
    let Reg_11_Red   = 13
    let Reg_11_Green = 12
    let Reg_11_Blue  = 14
    let Reg_12_Red   = 15
    let Reg_12_Green = 0
    let Reg_12_Blue  = 0
    let Reg_13_Red   = 15
    let Reg_13_Green = 7
    let Reg_13_Blue  = 0
    let Reg_14_Red   = 15
    let Reg_14_Green = 14
    let Reg_14_Blue  = 0
    let Reg_15_Red   = 0
    let Reg_15_Green = 0
    let Reg_15_Blue  = 0
end sub

sub Use_New_Colors
    set color mix (0) Reg_0_Red/16, Reg_0_Green/16, Reg_0_Blue/16
    set color mix (1) Reg_1_Red/16, Reg_1_Green/16, Reg_1_Blue/16
    set color mix (2) Reg_2_Red/16, Reg_2_Green/16, Reg_2_Blue/16
    set color mix (3) Reg_3_Red/16, Reg_3_Green/16, Reg_3_Blue/16
    set color mix (4) Reg_4_Red/16, Reg_4_Green/16, Reg_4_Blue/16
    set color mix (5) Reg_5_Red/16, Reg_5_Green/16, Reg_5_Blue/16
    set color mix (6) Reg_6_Red/16, Reg_6_Green/16, Reg_6_Blue/16
    set color mix (7) Reg_7_Red/16, Reg_7_Green/16, Reg_7_Blue/16
    set color mix (8) Reg_8_Red/16, Reg_8_Green/16, Reg_8_Blue/16
    set color mix (9) Reg_9_Red/16, Reg_9_Green/16, Reg_9_Blue/16
    set color mix (10) Reg_10_Red/16, Reg_10_Green/16, Reg_10_Blue/16
    set color mix (11) Reg_11_Red/16, Reg_11_Green/16, Reg_11_Blue/16
    set color mix (12) Reg_12_Red/16, Reg_12_Green/16, Reg_12_Blue/16
    set color mix (13) Reg_13_Red/16, Reg_13_Green/16, Reg_13_Blue/16
    set color mix (14) Reg_14_Red/16, Reg_14_Green/16, Reg_14_Blue/16
    set color mix (15) Reg_15_Red/16, Reg_15_Green/16, Reg_15_Blue/16
end sub

sub Wait_Response
    do
        if key input then
            get key k
            if k=324 then exit do
        end if
   loop

   clear
   set cursor 10,17
   print "... Press left mouse button to clear screen ..."
end sub

end
```

•AC•

Multi-Forth

A Designer Language Explored

by Lonnie Watson

Before I delve too deeply into this article, perhaps I should state that the code samples included with this article are written in Creative Solutions Multi-Forth programming environment. I selected Multi-Forth because, at the time, it was the only Forth development package available for the Amiga. There were a few shareware releases of Forth compilers around, but nothing compared with Multi-Forth's abilities and features.

I have since tried every Forth compiler on the market, but I have always returned to Multi-Forth as my main compiler. However, because most Forths adhere to some standards, it is not all that difficult to convert these code samples to work with the other Forth compilers available.

Forth is considered a designer language—it is amazingly flexible, allowing all manner of approaches to a problem. With this flexibility comes a certain lack of standardization that makes the language a bit difficult for a novice. Part of that difficulty stems from the fact that, by and large, the programming community has skipped over Forth in favor of the other more widely accepted languages, such as C and Modula-2.

However, anyone who has effectively programmed in Forth will attest to the insiders' view: it is perhaps the most natural of all programming languages. Combining the best of all programming worlds, Forth allows the programmer to concentrate on the real problem at hand rather than struggling with the operating system they may be surrounded by.

In the Amiga environment, Forth is especially powerful. In my case, Forth has allowed me to perform programming feats with my Amiga that I never found possible in other languages. One problem I have encountered while dealing with Forth is the lack of code samples that perform mundane functions that are perhaps STANDARD macros on other languages, such as already-defined structures that implement such system-dependent data types as Windows and Screens. Even the interfaces with Amiga-specific functions in ROM or on disk are left to the programmer to implement. This type of side-tracking often stalls a programmer, resulting in a graveyard of unfinished business.

In this series of articles, I will attempt to show ways of dealing with some of these problems. First, we will take a look at how to implement an interface to the much-talked-about ARP library.

What? No ARP?

What's that you say—you don't have the ARP library? Where have you been for the past year or two? ARP is a library of routines designed to replace the DOS library built into Kickstart . This DOS library, on disk for A1000 owners and in ROM for A500 and A2000/2500 owners, has several quirks that make it difficult to work with on a very low level. The ARP library actively replaces most, if not all, of the functions that the DOS library has built in, while supplying some nice extras. Two of my favorites are the ARP FileRequest routine and the ARP StringCompare routine (with full pattern matching). These functions go a long way in preventing that damaging side-tracking that often kills program development.

In order for these sections of code to work, you will need the ARP library in your LIBS: directory. If you don't have ARP, you can get it from most local BBS's or on Fred Fish disk #123.

As with most tasks on the Amiga, interfacing with a library from Forth requires a certain number of predefinitions and other housekeeping. Forth allows easy manipulation of libraries, and even automatically closes all libraries that are left open at the exit of your program. We will define a word that will open the ARP library and place it as library 16 in the Multi-Forth system. First, we will define some constants that will clarify the library-opening process.

```
0    CONSTANT ARPLIBREVNUMBER      \ rev necessary
16   CONSTANT ARP-FORTHLIB         \ actual lib # in forth
256  CONSTANT ARPBUFMAX            \ MAX length of names

\ The following word will open the ARP library provided it is in
\ the LIBS: directory and place it as lib # 16 in the Multi-
Forth
\ system:

: OPEN_ARP ( – ) \ ARP opened as lib 16 or abort if unable
0" arp.library" arplibrevnumber arp-forthlib open.lib
not if abort then ;

\ note the case of the zero delimited string. This must match or
\ the open.lib word will be unable to find the library. This is
\ in stark contrast with the rest of the system where generally
\  case is of little consequence.
```

Being a library of routines, ARP has certain requirements that differ depending on what routines you want to use. The first library call we will examine will be the FileRequest routine built into the Library. As with most Amiga Library functions, the ARP FileRequest function requires a structure definition from which it obtains the necessary data to function. The ARP FileRequest structure, as it is implemented in Forth, looks like this:

```
STRUCTURE ARPFILEREQUEST
 LONG:     +ARPGREETMSG        ( ptr to titletext )
 LONG:     +ARPFILEBUF         ( ptr to filename buffer )
 LONG:     +ARPDIRECTORY       ( ptr to DIRname buffer )
 LONG:     +ARPWINDOW          ( ptr to screen for )
                                 ( arp window. NULL = WB )
 BYTE:     +ARPFUNCFLAGS       ( see below for meanings )
 BYTE:     +ARPRESERVED1       ( paddind for aligmnent )
                                 ( set to NULL )
```

```
 LONG:     +ARPFUNCTION        ( ptr to call for wildcards )
 LONG:     +ARPRESERVED        ( rsvd for future upgrades )
 STRUCTURE.END

 \ Bit meanings for the +ARPFUNCFLAGS function
 \ BIT#     MEANING
 \   0       List function. Not implemented prior to ver 34
 \             of the library.
 \   1       user gadgets enable bit
\    2       added gadgets enable bit
\    3       allows modification of new window structure
\    4       new IDCMP only if +ARPWINDOW is NULL
\    5       Set this bit for different color scheme
\    6       Pass IDCMP messages not intended for file request
\    7       Do wildfunction flag ???
```

Now that the structure is defined, we need to create an actual instance of that structure in memory. All we have done so far is create the template from which we will obtain the various fields in the structure once we have the structure created. In Multi-Forth, creating an actual instance of a structure such as this one is very easy:

```
STRUCT ARPFILEREQUEST ARPFILEREQUEST1 STRUCTEND
```

Here we have created an actual instance of an ARPFILEREQUEST structure and named it ARPFILEREQUEST1. When the word ARPFILEREQUEST1 is encountered either in the input stream or in a definition, the address of the structure's first byte will be left on the Forth parameter stack. Notice that the structure contains a number of pointers to buffers where data will be placed. These buffers must be allocated and can be preset with default parameters that will show up in the actual file request window. Parameters like the default Path name and the default filename can be selected this way. The buffers can be defined as such:

```
VARIABLE  ARPFILEBUF1    ARPBUFMAX ALLOT
VARIABLE  ARPDIRBUF1     ARPBUFMAX ALLOT
```

Now, a word that will initialize the FileRequest structure and place pointers to the buffers in the structure:

```
: INIT_ARPFILEREQUEST ( - )
ARPFILEREQUEST1 ARPFILEREQUEST 0 FILL
ARPFILEBUF1 ARPBUFMAX 0 FILL
ARPDIRBUF1 ARPBUFMAX 0 FILL
\use this code line if you have a custom screen open and want
\the file request to show up on your screen. Otherwise it will
\appear on the Workbench screen:
\ CURRENTSCREEN @ ARPFILEREQUEST1 +ARPWINDOW !
ARPFILEBUF1 ARPFILEREQUEST1 +ARPFILEBUF !
ARPDIRBUF1 ARPFILEREQUEST1 +ARPDIRECTORY !
0" SAMPLE TITLE TEXT " ARPFILEREQUEST1 +ARPGREETMSG ! ;
```

Now that this is defined, all we need to do is create a word that will actually call the ARP library FileRequest function with the initialize structure as a parameter. This word will also have to handle the return value that the FileRequest function passes back to the program, and take action. For the time being, our definition will only respond to the return of a NULL. If you receive a NULL from the FileRequest function, you have selected the cancel gadget. The definition of the calling word looks like this:

```
: DO_ARP_FILEREQUEST ( - flags )
ARPFILEREQUEST1 !a0 call.lib 16 49 @d0 255 and ;
```

By using the 255 and phrase at the end of the definition, we have masked off the upper 24 bits of the long word that is placed on the stack from the CPU register D0 after the library call. Since Multi-Forth only allows pulling long words from registers, we need to mask off the unwanted bits to make sure we have the right number as a return value.

Now we are ready to use these words to actually get a FileRequest on the screen. A sample invocation might look like this:

```
: DOIT
open_arp
init_arpfilerequest
do_arp_filerequest
```

The previous example demonstrated the use of the ARP FileRequest function. By using this function, users can quickly implement a fast and efficient FileRequest routine into their own programs. Program development can then proceed in a timely fashion, since the programmer can concentrate on what is important in the code, not on such a trivial section of code as a File Requester.

The next ARP function that we will look at is actually a small set of functions designed to perform one task—to perform string compares with full pattern matching. Pattern matching allows for the use of wildcards in string compares. Often in a program, a need arises to compare two zero-delimited strings of characters. In database programs, for example, string comparisons occur all the time, especially when the user is searching through a database for a particular set of data.

Pattern matching allows even greater power and flexibility when the user performs searches. For example, say the user wants to find someone on their mailing list, but does not know the complete first or last name. All the user knows is that the person's last name begins with the letters ABER. Using pattern matching, the user would key in ABER* and the string search routines in the program would find all the people who had ABER as the first four letters of their name. Such a search would bring up names such as ABERCROMBIE, ABERLONIAN, and ABERNONE. This sort of searching is often difficult and time-consuming to program, so why bother? The ARP library has these functions built right in.

The implementation of the string compare with pattern matching is actually distributed among several routines. The first routine compares two strings and returns values based on that comparison; there is no pattern matching going on here. The string compare word is implemented as follows:

```
: DO_ARP_STRCMP ( 0$a 0$b - a<b -1,a=b 0,a>b 1 )
\ addr of two zero delimited strings left on the stack
\ returns 0 if strings are equal. Case ignored
\ returns -1 if string 1 is less than string 2. Case ignored
\ returns 1 if string 1 is greater than string 2. Case ignored
!a1 !a0 call.lib 16 43 @d0 ;
```

Using this word is fairly straightforward. All that is required is that the ARP Library must be open already as library 16 (the open_arp word from above is called first). Once the ARP Library is open, the string compare word can be invoked like so:

```
\typed in directly  at the Forth console
```

```
0" Hi" 0" By" DO_ARP_STRCMP . <return>

\will print a 1 on the screen.
```

The user can play with this word by supplying his own zero quote strings and calling the String Compare word directly from the keyboard. In doing so, it is easy to imagine various ways to use this little gem.

But, I can hear you out there yelling, "HEY! What about pattern matching?! Why would he make me read all that verbiage about pattern matching?!" Well, hang on one moment because here's the pattern-matching code.

As stated earlier, the pattern-matching code is separated into two different routines. The first routine takes a zero-delimited string containing the wildcard pattern to be searched for, and converts it to a tokenized partial ASCII zero-delimited string which the subsequent pattern match routine can use to search for a regular string (zero-delimited, of course). Stated another way, the Preparse function is the food processor for the pattern, and the patternmatch is the consumer of the processed pattern.

```
\first we will define a buffer that will get out pattern
\and that our preparse routine will deposit the parsed pattern
\back into.

VARIABLE ARPPARSEDBUF1 ARPBUFMAX ALLOT

\next we will implement the actual preparse word.

: DO_ARP_PREPARSE ( 0$ADDR to be parsed — true if wild found )
\the routine will place its parsed result in the string
\arpparsedbuf1 after it clears the buffer with zeros
arpparsedbuf1 arpbufmax 0 fill
arpparsedbuf1 !a1 !a0 call.lib 16 93 @d0 ;
```

Now we will define the actual pattern match word. This word will take a zero-delimited string whose address is on the stack, and use the assumed preparsed string in arpparsedbuf1 as a key of what to look for. The result will be true if there is a match and false otherwise:

```
: DO_ARP_PATTERNMATCH ( 0$addr to look at — true or false )
arpparsedbuf1 !a0 !a1 call.lib 16 72 @d0 ;
```

Using these functions is as easy as passing the wildcard that you want to look for to the DO_ARP_PREPARSE function, then passing the string to be searched to the DO_ARP_PATTERNMATCH function. Note that the DO_ARP_PREPARSE function returns a true if there was actually a wildcard in the parsed string. If the function returns a false, then there was no wildcard in the parsed string, and the software can then use the false value to trigger the regular string compare function, which is faster.

That's all there is to it. Imagine the effort saved by not having to create code to perform these functions. There is a whole host of other goodies in the ARP Library which can save you many hours of typing at the keyboard. We will look at these functions at another time.

•AC•

More Requesters in AmigaBASIC

by John R. Wiederhirn

Up to now, the articles in this series have covered programming using AutoRequesters and Alerts from within BASIC. While the information they contain is applicable to certain situations, there may be times when we'll want to do something as simple as putting up a message with an "O.K." button. To do this with an AutoRequester, you would have to give the message, give both buttons the same text, wait for a response, and ignore which button the user actually clicked on. While it could be done with an Alert, such an approach would be highly impractical.

If you wanted to offer the user more than two choices, neither of the previous methods would do. Ordinarily, that would have meant having to go back to doing your own requesters from BASIC. Not anymore. The routines in this article will allow you to set up a requester with as many or as few selections as you need, as well as give you much greater flexibility in the appearance of the Requester and its buttons.

The general theme of the previous articles has been how to use system functions to allow BASIC programs to do things which would normally fall exclusively within the realm of C and other developer-oriented languages. This article continues that theme.

To generate and get the response to the requesters, a small assembly language subroutine is read in and called by BASIC. This routine provides a "courier" service between Intuition and BASIC. You will not need an assembler or any other special tools to enter the routine, but you will need to type in a couple of lines of BASIC data statements which contain the assembled code that BASIC reads in and calls.

BASIC has tremendous flexibility in controlling the system, but it does lack a facility to easily receive the messages from Intuition. Those messages are used to indicate when Gadgets—buttons, in your case—have been pressed, and when other system events—above and beyond mouse movements and menu selections—have occurred.

Like those in the previous articles, the routines used in this program are designed to be "snipped out" and used in your own programs. Just enter the listing, save it, and then run it once or twice. The routines are fully explained below, as are methods on how to use them in your own programs. Before I can explain exactly what's going on in the program, however, we'll need to understand a number of new system structures that are necessary to make working requesters. If you are already comfortable working with Gadget, Border, NewWindow and Requester structures, feel free to skip ahead. For the rest of you, let's get to it.

The Supporting Characters

I'm going to assume that you have some familiarity with the topics covered in the previous articles, so I will not repeat explanations of the IntuiText structure or the memory management system routines (AllocMem, FreeMem, and CopyMem). There is a whole new set of system structures that are used along with IntuiTexts to create requesters, and some rules that go with setting up each one.

First, let's start with the most basic structure after the IntuiText: the Border structure. The system uses Borders to create borders around gadgets, requesters, and other Intuition objects. Each Border structure describes a continuous set of line segments in a single color. To generate a blue and orange outline around a space, you would need two linked Border structures. This article is only going to deal with single-color borders, so the linking will be left for some other time. The C language definition of a border is as follows:

```
struct   Border   {
short    LeftEdge, TopEdge;
BYTE FrontPen, BackPen;
BYTE DrawMode;
BYTE Count;
short    *XY;
struct   Border   *NextBorder; };
```

Since the Amiga was designed for C programs to use the Intuition routines easily, the system uses structures quite a bit. For BASIC programmers, however, there is no real equivalent. As before, the easiest way to work around this problem is to allocate the proper amount of memory using AllocMem, and then poke the data for each field into memory, creating a "virtual" structure which the system can use. To do so, you need the offsets and type of each field in the structure. In the future, these will follow each structure's C definition. Here are the offsets and types for the border's fields (more on what they mean coming up):

```
Field        Offset   Type
LeftEdge     +0       short integer
TopEdge      +2       short integer
FrontPen     +4       byte (0-255 int)
BackPen      +5       byte
DrawMode     +6       byte
Count        +7       byte
XY           +8       pointer (long int)
NextBorder   +12      pointer
```

LeftEdge and TopEdge will be set to -1 for your purposes, telling Intuition to use the default settings (usually 0). However, they still refer to the X and Y coordinates for the origin of the XY pairs, relative to the Intuition object of which the border is a part. More on that in a second.

The FrontPen and BackPen fields select which of the color registers (0-3) will be used to draw the border and its background. The color set in BackPen shows only for dotted lines, and since solid lines have no background, you will set the color

to 0 for your uses. DrawMode serves the same purpose here as it does in an IntuiText. You will use JAM1 mode for all borders and IntuiTexts, but that doesn't mean you couldn't use others.

Count and XY serve special purposes. Count is the number of XY pairs that form the border. For a rectangular border, count is set to 5. XY is a pointer (long integer address) to a series of pairs of short integers (first X, then Y) which are the corners of the border lines. The line you are describing will be continuous, so the last pair is the same as the first. For example, a border box at (0,0) that was 100 pixels wide and 30 pixels high would have a Count of 5, and *XY would point at 10 short integers (each two bytes long) with the following values:

```
0, 0, 100, 0, 100, 30, 0, 30, 0, 0
```

These values describe the four corners of the box, and ensure that it's a closed rectangle. Incidentally, the values for each X and Y coordinate are relative to the LeftEdge and TopEdge values of the Border structure, which are in turn relative to the LeftEdge and TopEdge of the Gadget or Requester structure of which the Border structure is a part. To clear it up a little, suppose a Requester was set with a LeftEdge and TopEdge of (0,0). The Border structure for the requester would have a LeftEdge and TopEdge of 5 (5,5), and the first XY pair would be 3,3. The actual location on the screen would be

```
(0+5+3), (0+5+3), or (8,8).
```

Last, and for you, least important of the Border fields, the NextBorder field is a pointer to the address of the next Border structure in a list, and is used to create the "linked" multi-color borders discussed above. Just set it to NULL (0& in BASIC).

The next structure used to create requesters is the Gadget structure. It contains the definition of a "button" in the requester, including links to IntuiTexts (which hold the text of the button) and borders (for the outline of the button). The C definition and offset table for a Gadget structure are as follows:

```
struct   Gadget   {
struct   Gadget   *NextGadget;
short    LeftEdge, TopEdge;
short    Width, Height;
short    Flags;
short    Activation;
short    GadgetType;
APTR     GadgetRender;
APTR     SelectRender;
struct   IntuiText    *GadgetText;
long     MutualExclude;
APTR     SpecialInfo;
short    GadgetID;
APTR     UserData; };
```

```
Field       Offset Type
NextGadget    +0   pointer (long int address)
LeftEdge      +4   short int
TopEdge       +6   short int
Width         +8   short int
Height        +10  short int
Flags         +12  short int
Activation    +14  short int
GadgetType    +16  short int
GadgetRender  +18  pointer
SelectRender  +22  pointer
GadgetText    +26  pointer
MutualExclude+30   long int
SpecialInfo   +34  pointer
GadgetID      +38  short int
UserData      +40  pointer
```

The fields of the Gadget structure afford tremendous flexibility in the design of the "buttons" for a requester. Much of that flexibility, however, is beyond the scope of this article. You need to concern yourself only with the values and settings which are used to construct a simple button consisting of text within a border. If you're inquisitive, you can consult either the Amiga Intuition Manual or one of the many other books and articles which cover the possibilities available using gadget construction.

The first field is NextGadget, which is a pointer to another Gadget structure and is used to link lists of gadgets within windows, requesters, etc. When you create each gadget you set that field to 0&. Another section of the program will "link" the various gadgets to form the list. All that will be covered under the description of the program.

Next, you have the usual LeftEdge and TopEdge fields. These locate the upper-left corner of the gadget relative to the window or requester in which it is located, as discussed above in the border explanation. Remember, the Border structure's LeftEdge and TopEdge are relative to these figures, which in turn are relative to the LeftEdge and TopEdge of the requester.

Width and Height describe the "hot" region of the gadget, where clicking the mouse will select that gadget—and, for you, return a selection message.

Flags contains the Intuition code that selects how the gadget is highlighted when selected. You will set this to the value for GADGHCOMP (0), which tells Intuition to highlight the gadget by "complementing" the hot region. In other words, since your gadgets use black lines and text on a white background (under standard Workbench colors), highlighting is achieved by showing white lines and text on a black background.

The Activation value controls which message is sent by Intuition when the gadget is selected. You set this so that the assembly language subroutine receives a message as soon as the Gadget is "clicked"; this is the GADGIMMEDIATE setting (2). As soon as the Gadget is selected, Intuition sends a GADGETDOWN message to the window in which the gadget is located, telling it that a gadget was selected, and which one it was (via GadgetID).

GadgetType tells Intuition exactly what kind of gadget this structure is—information which affects Intuition's handling it. Your gadgets are all simple push-buttons and are located within requesters. Thus, you will use the value found by boolean OR'ing the settings for REQGADGET (which tells Intuition that the gadget is in a requester) and BOOLGADGET (which tells Intuition that the gadget is a push-button): 4097 (&h1000 OR &h0001).

GadgetRender and SelectRender control what Intuition displays for the gadget in its non-selected and selected states. If you were using a bitmapped gadget, they would point to structures for the appropriate bitmaps. Since you are not, and have set a "preset" way of showing selection (GADGHCOMP), set SelectRender to 0&. The GadgetRender field serves another purpose besides pointing to a bitmap image; it can point to a Border structure when a non-image gadget is being used—as it is in this case. By setting GadgetRender to point at a Border structure for each button, a much more attractive button is drawn (rather than text floating in space).

Speaking of text, the GadgetText field points to the IntuiText structure that defines the text for each button. This is the same IntuiText structure you may have come to know in the

previous articles. Within the scope of this article, you will set the MutualExclude, SpecialInfo, and UserData fields to 0& in your gadget definitions, as your buttons do not need them. Again, for more information on the different settings of gadget fields, consult an article or book which covers Intuition programming.

The only field left to cover is the GadgetID field, and it is an important one. This short integer is the code Intuition sends (indirectly) to the Window containing the gadget, in order to indicate which gadget was responsible for the GADGETDOWN message. It doesn't directly send the GadgetID as part of the message (called an IntuiMessage), but instead sends a pointer to the Gadget structure of the button selected. The assembly language routine then grabs the GadgetID of that structure and returns that value to the BASIC program. GadgetIDs must be unique to each Gadget structure, unless you don't really want to know which one was selected. I have run across a case or two where you wouldn't, but "nice" gadgets should have unique GadgetID numbers.

The next system structure of concern is the Requester structure. This structure contains the definition of the area in which all the buttons appear and connects to the linked list of gadgets to appear inside the requester. Here are its C definition and offset values:

```
struct   Requester    {
struct   Requester    *OlderRequest;
short    LeftEdge,    TopEdge;
short    Width,       Height;
short    RelLeft,     RelTop;
struct   Gadget       *ReqGadget;
struct   Border       *ReqBorder;
struct   IntuiText    *ReqText;
short    Flags;
BYTE     BackFill;
struct   Layer        *ReqLayer;
BYTE     ReqPad1[32];
struct   BitMap       *ImageBMap;
struct   Window       *RWindow;
BYTE     ReqPad2[36]; };
```

```
Field       Offset Type
OlderRequest +0  pointer
LeftEdge     +4  short int
TopEdge      +6  short int
Width        +8  short int
Height       +10 short int
RelLeft      +12 short int
RelTop       +14 short int
ReqGadget    +16 pointer
ReqBorder    +20 pointer
ReqText      +24 pointer
Flags        +28 short int
BackFill     +30 byte
ReqLayer     +32 pointer
ReqPad1[32]  +36 byte array
ImageBMap    +68 pointer
RWindow      +72 pointer
ReqPad2[36]  +76 byte array
```

Okay, I admit that it is definitely not one of the smaller structures on the Amiga. There is an up side, however. You see, to create a Requester structure in C, you declare the structure. But before you set any of the values, you call a system routine called InitRequester. Its sole purpose is to clear all the bytes in a Requester structure. You can use this routine from BASIC to do the same thing. After doing this, you need to set only the fields which relate to your needs. For you, that means setting LeftEdge and TopEdge, Width, Height, ReqGadget, ReqBorder, ReqText, and BackFill. While the other fields allow some wonderful

flexibility, you just don't need them for what you will be doing. For this reason, I will explain the functions only of those fields you are going to use.

The InitRequester routine takes a pointer to (the address of) a Requester structure, and returns no values to BASIC. The calling syntax looks like this:

```
CALL InitRequester&( reqAddress& )
```

The reqAddress& variable would be the address of the block of memory you are calling a Requester structure.

LeftEdge and TopEdge function the same here as in the other structures—in this case, as X- and Y-Offsets from the upper-left corner of the window containing the requester. For technical reasons explained below under the NewWindow structure, these will be set to 0 for the requester, but operate the same here as they do for borders and gadgets.

Width and Height are essentially the same fields you are used to, with minor differences. Instead of pointing to a hot region, they define the outer boundaries of the requester box, and also set the size of the area covered by the requester's BackFill.

ReqGadget is a pointer to the first in a list of Gadget structures. This means that after you define all the gadgets, you link them together and put the address of the first one here. The last gadget in the list has a 0& in its NextGadget field, which tells Intuition there are no more gadgets in the list.

ReqBorder is a pointer to the Border structure you use to outline the requester's box. There is one major difference between this border and the gadget borders discussed earlier:

While the gadget border XY pairs are true outlines—in that they start with an upper-left corner of (0,0)—the requester border must be inside the outer edge of the requester. Accordingly, you set it with an upper-left corner of (5,3) and keep an X-Offset of 5 and a Y-Offset of 3 from the outer edge of the entire border.

ReqText, as you probably guessed, is a pointer to the IntuiText that contains the message for the requester itself. Although you won't do so in this program, you can generate multi-line text messages by linking IntuiTexts in the requester and gadget structures in the same manner you did in AutoRequesters. (Here you just use a simple one-line message to tell the user what to do.)

Finally, BackFill is the register number (0-3) of the color you want to be the background for the requester. I set that to white (1), but this could easily be changed. One interesting note: By setting the BackFill to 0, Intuition can generate a "transparent" requester, where the background is the "surface" of the window behind the requester. Here, however, you use a special window, so there is nothing to see. Since the assembly language routine creates and deletes the window, you have no way to give it a background worth looking at, anyway.

That brings me to a rather important point. The final structure you use is the NewWindow structure, which Intuition uses to define a window before opening with OpenWindow. Normally, this would not be a required part of a requester, which could float on the current window in use. Unfortunately, AmigaBASIC does some naughty things with the windows that it opens; as a result, even an assembly language program would have to do some serious gymnastics to receive IntuiMessages from a window opened by BASIC.

There is a much simpler solution: If windows opened by BASIC aren't usable, open one for yourself. The result is that the assembly routine needs to be passed a NewWindow structure, which it uses to open a window in which the requester is placed. The definition of the window to be created is set by the program based on the size of the requester, since the window exists solely to contain the requester, and ceases to exist once the requester is finished.

A nice advantage to using a separate window is that it can be dragged around and moved to a more convenient location if it happens to cover important data. There is, however, a catch. Because of the peculiar way in which BASIC calls assembly and system routines, it is conceivable that while a requester is being displayed in its window, BASIC could be terminated. It is also possible that responding to the requester in such a situation could crash the system—or, at the very least, maim it.

The assembly routine is not bulletproof, and can probably be tricked into doing dangerous things. The best solution would be somehow to prevent BASIC from being terminated during program execution; unfortunately, there is no readily available way to do this. The next best thing is to foolproof your program as much as possible. After all, terminating BASIC while running a program can cause unpredictable results, even without system routines being used. Save once, save twice, and if you still aren't sure, save it again!

On with the discussion of system structures... The last structure you need specs for is the NewWindow structure mentioned earlier. The definition and offsets of the NewWindow structure are as follows:

```
struct   NewWindow    {
short    LeftEdge,    TopEdge;
short    Width,       Height;
BYTE     DetailPen,   BlockPen;
long     IDCMPFlags;
long     Flags;
struct   Gadget       *FirstGadget;
struct   Image        *CheckMark;
BYTE     *Title;
struct   Screen       *Screen;
struct   BitMap       *BitMap;
short    MinWidth,    MinHeight;
short    MaxWidth,    MaxHeight;
short    Type; };
```

Field	Offset	Type
LeftEdge	+0	short int
TopEdge	+2	short int
Width	+4	short int
Height	+6	short int
DetailPen	+8	byte
BlockPen	+9	byte
IDCMPFlags	+10	long int
Flags	+14	long int
FirstGadget	+18	pointer
CheckMark	+22	pointer
Title	+26	pointer
Screen	+30	pointer
BitMap	+34	pointer
MinWidth	+38	short int
MinHeight	+40	short int
MaxWidth	+42	short int
MaxHeight	+44	short int
Type	+46	short int

Even though you don't have a nice system routine like InitRequester to clear out a NewWindow structure, set to 0 all the bytes in the memory you allocate for it. That way, all you'll need to do is set the fields that are necessary for your use. For the window to work with the assembly routine, you need to set LeftEdge and TopEdge, Width, Height, DetailPen, BlockPen, IDCMPFlags, Flags, Title, Min- and Max- Width and Height, and Type. The rest you can just ignore. Again, I will explain only the fields that you will actually use.

The LeftEdge and TopEdge values given here are actual, honest-to-goodness screen coordinates. These are the master values to which the requester—and, indirectly, gadget and border—fields of the same name are relative. You will set these as the upper-left corner of the window in which you want the requester to appear on the screen.

Width and Height mean pretty much the same thing as they do in the Requester structure, referring to the size in pixels of the actual graphic image produced by Intuition. As you shall see later in the program, you set these values based on the size of the requester to be put in the relevant window.

You will set DetailPen and BlockPen to generate a window image that conforms to the standard Intuition window—white graphics with blue outlines and details. DetailPen gets set to 0 (blue) and BlockPen gets set to 1 (white), to create properly colored graphics.

The IDCMPFlag's field tells Intuition what kinds of IntuiMessages you want this window to receive. Since all you need to receive is GADGETDOWN messages, set IDCMPFlags accordingly (32&). By setting other bits, you can get different kinds of messages, but GADGETDOWN is the only kind that the assembly routine understands; it ignores everything else.

Flags—not to be confused with the previous field—define window generation for Intuition. Set it as ACTIVATE,

SIMPLE_REFRESH, WINDOWDRAG, and GIMMEZEROZERO. In English, this means that you want the window to activate when it first appears; you don't have a background to save, so there's no need to bother. You'll want to be able to drag the window (and the Requester, as well) around the screen. Also, you'll want the upper-left corner of the "usable" (non-gadget) part of the window to have the coordinates (0,0). You ensure that they do by setting the appropriate bits in the Flags field, using a value discussed later in the program section.

Note: According to the ROM Kernel Manuals, there are still ways for the user to prevent the window from becoming the active window when opened, even if ACTIVATE is used. As usual, if the user really wants to mess things up, he/she probably can. I feel obligated to try to follow Commodore doctrine, if only to prevent yet another source for program errors.

If you want the window holding the requester to have a title, you just set the Title pointer to the address of a string of characters holding the title and ending with a CHR$(0). I think it makes the requester appear too "busy," but set it to 0& (meaning no title). This setting makes the drag bar expand to fill the space where the title would have been.

All the Min's and Max's are for resizable windows. Since you don't have to use such a window, you can set them to the same values you assigned Width and Height.

The last field of importance in the NewWindow structure is the Type field. This tells Intuition whether the window is for a Workbench screen or a custom screen. We're calling this from BASIC, which normally has a Workbench screen, so this field gets set to 1 (Workbench screen). The requesters generated and used by these routines must be on a Workbench screen to function correctly. Normally, the assembly routine opens the window, using the OpenWindow system routine, and does not expect to have to link the routine to a custom screen. Trying to put a requester on a custom screen will result either in the requester and window being put on the Workbench screen, or in a visit from you-know-who.

That, in a nutshell, is what you need to know about Intuition Requesters. Once you understand how the program works, feel free to use your new knowledge to tinker around with the setting of these structures. Remember the various warnings discussed above, however; nobody likes programs that crash frequently. Now let's get to the explanation of how the demonstration program works.

The Lead Roles (or How It Works...)

As you go through each of the routines, learning how they work and relate to each other, you may want to flip back to the structure offset tables to see exactly what each POKE does. (The long blocks of POKE statements will not be shown in their entirety. Consult the program listing for the exact POKEs and values.) Since most of the stuff has already been covered in the other articles, I will spend little time explaining old material.

```
CLS
DECLARE FUNCTION AllocMem&() LIBRARY
LIBRARY "exec.library"
LIBRARY "intuition.library"
```

As usual, you have here the standard starting code, which clears the screen and opens the necessary libraries. Don't forget,

the exec.bmap and intuition.bmap files must be in either the current directory or the libs: directory of the system disk.

```
DIM mp&(4), msiz%(4)
DoReqst& = 0&
nam$ = "intuition.library"+CHR$(0)
dloc& = 0&
```

Moving on through the code of the main program, you initialize some of the variables that will be used later, and make nam$ a string ending in CHR$(0) and holding the name of the Intuition system library. This is very important, because the assembly routine needs that string to properly open Intuition for its uses. By the way, if everything else seems to work fine, but the assembly routine just returns without doing anything, make sure the CHR$(0) is added to the end of nam$.

```
LOADSBR DoReqst&, dloc&
SUB LOADSBR ( sptr&, dptr& ) STATIC
```

Here you start to get into the important routines. The LOADSBR routine is what actually loads the assembly routine into memory and returns two pointers. The sptr& parameter (here going into DoReqst&) tells where the assembly routine starts. DoReqst& is later used in the calling of the assembly routine. The dptr& parameter (here going into dloc&) gives you the location of a spare piece of allocated memory which is used in the program as storage for the result returned by the assembly routine.

```
MEMALLOCATE sptr&, 162&
```

Inside the LOADSBR routine, you call the MEMALLOCATE routine in preparation for loading the assembly routine into memory. All the MEMALLOCATE routine does for you is take the number of bytes needed, call the system routine AllocMem&, make sure everything went okay, and return a pointer to the allocated memory. I have made it a free-standing routine here because it is frequently used by the other routines. The MEMFREE is a similar case, existing only to call the FreeMem& system routine. I needed a little more flexibility in using FreeMem&, however, so MEMFREE is slightly more flexible than previous incarnations. Both of these routines have been covered before, so check the previous articles for explanations of their functioning.

```
dptr& = sptr& + 160
RESTORE MLData
```

Having the memory location where the assembly code is going, and knowing the length of that code, you can figure where your "spare" memory—which holds the data—starts. In this case, the location is 160 bytes beyond the start of the assembly code. You next use RESTORE to set the start of the DATA statements to the beginning of your code data (in case your program needs to use RESTORE for data of your own).

```
FOR i& = sptr& TO (sptr& + 158) STEP 2
    READ op%
    POKEW i&, op%
NEXT i&
POKEW dptr&, 0
```

These lines read the assembly code from the data statements and then poke them into memory a word at a time. After doing this, the spare memory to which the data pointer points is cleared. That's the end of the working part of LOADSBR.

```
MLData:
DATA &h48e7, &hfffe, &h2c78, &h0004, &h701d ...
END SUB
```

The remainder of the subprogram holds the assembly language routine in the data statements a word at a time in hexadecimal format. You must make sure these lines are typed in correctly for the program to work. After the data statements, the END SUB returns execution to the main program.

```
FOR i% = 1 TO 3
  b% = i%
  by% = 1 + ( i% * 15 )
  bt$ = "Choice" + STR$(i%)
  MKBTN 10, by%, 240, 12, 2, 5, 2, bt$, b%, bp&

SUB MKBTN (x%,y%,w%,h%, c%, tx%,ty%,t$, id%, bp& ) STATIC
```

The requester being made by the program will have three buttons, with text in each reading "Choice 1," "Choice 2," and "Choice 3." The FOR...NEXT loop sets up, creates, and links the three buttons. Here the program initially sets up the parameters for the buttons, calling MKBTN for each. MKBTN takes the values needed for the Gadget, Border, and IntuiText structures (x% - id%), and returns a pointer (in bp&) to the location of the Gadget structure it created in memory. It also returns the length of the IntuiText structure (the structure itself, plus the text data) in id%, which you later use when freeing the allocated memory.

```
MEMALLOCATE bp&, 80&
xy& = bp& + 44
bdp& = xy& + 20
```

Inside MKBTN, the routine first allocates enough memory for the Gadget and Border structures, as well as for the XY data for the border. It then uses the pointer returned by MEMALLOCATE as the start of the Gadget structure and calculates pointers to the XY data and Border structure based on their lengths. The xy& pointer is the start of the XY data, while the bdp& pointer is the start of the Border structure.

```
MKITEXT tx%, ty%, 2, 0, 0, t$, itp&, its&
SUB MKITEXT ( x%, y%, fc%, bc%, m%, t$, itp&, its& ) STATIC
```

MKBTN then passes on the appropriate data to the MKITEXT routine to create an IntuiText structure to hold the button's text. The MKITEXT routine is virtually the same as the CreateIText routines in the previous articles; it has simply been revised to use MEMALLOCATE and to place the text data immediately after the IntuiText structure, in the same block of allocated memory. I did this to help prevent memory segmentation, and to speed things up a little. The MKITEXT routine returns a pointer to the IntuiText structure, as well as the size of the memory block that was allocated. This makes freeing the memory later much easier. Check the other articles for more information on how to set up IntuiTexts from BASIC.

```
POKEL bp&,      0&
through
POKEL bp& + 40, 0&
```

This batch of POKEs actually puts the proper values into the Gadget structure you "faked" in memory. The locations are found by using the known offsets from the pointer you established as the start of the Gadget structure.

```
x1% = 0
x2% = w%
y1% = 0
y2% = h%
POKEW xy&,     x1% : POKEW xy& + 2, y1%
through
POKEW xy& +16, x1% : POKEW xy& +18, y1%
```

Here you create the XY data you need in memory to generate the proper border graphic. You calculate which two X and Y values to use, and then proceed to arrange them to create the box outline you want.

```
POKEW bdp&,     -1
through
POKEL bdp& +12, 0&
```

Finally, MKBTN puts the needed information for the Border structure into memory. Again, you use your pointer and the known offsets to set the data into memory correctly.

```
id% = its&
END SUB
```

Because you already have the bp& variable set to pass the pointer of the Gadget structure back to the main program, you just set id% to the length of the IntuiText structure and return to the main program.

```
    mp&( i% ) = bp&
    msiz%( i% ) = b%
    IF ( i% > 1 ) THEN
      bp1& = mp&( i%-1 )
      LINKBTN bp1&, bp&
    END IF
NEXT i%
```

Back in the main program, you put the pointer to the button's memory block and IntuiText size into arrays for later memory accounting. Then, for buttons two and three, you call LINKBTN to connect them so that button one is at the start of the linked list and button three at the end. All LINKBTN does is replace the NextGadget field of the button pointed to in its first parameter with the pointer given as the second parameter. You don't have to keep LINKBTN as a separate routine (it could be replaced with a POKEL statement), but it keeps things coherent in this program. Feel free to replace it if you need extra speed or smaller code size in your own programs. All you have to do is use the following line (replacing bptr1& and bptr2& with the proper variables from your program):

```
POKEL bptr1&, bptr2&
```

This line of code is all that LINKBTN consists of. However, separating it into a different routine does improve readability a little. The button creation loop finishes with a NEXT statement.

```
rx% = 30
ry% = 30
bptr& = mp&(1)
rt$ = "Make Your Selection:"
MKREQ rx%, ry%, 260, 64, bptr&, rt$, rp&, wp&

SUB MKREQ ( x%, y%, w%, h%, bp&, t$, rp&, wp& ) STATIC
```

The lines before the call to MKREQ simply set up the parameters for MKREQ to create the Requester structure. MKREQ also creates the NewWindow structure, which has a couple of fields that are calculated based on the figures for the Requester structure. MKREQ returns three values in rp&, wp&, and x%. The rp& value is a pointer to the start of the Requester structure in memory. The wp& value is the same thing, only for the NewWindow structure. The routine returns the size of the IntuiText structure of the requester in x%, for memory accounting.

```
MEMALLOCATE rp&, 196&
CALL InitRequester&( rp& )
rxy& = rp& + 112
rbp& = rxy& + 20
wp& = rp& + 148
```

First, MKREQ allocates memory for the requester, NewWindow, border, and XY data in memory, and sets pointers to the start of each area. It also calls the system routine InitRequester to clear out the area for the Requester structure. While this could just as easily be done using a FOR..NEXT loop—as when clearing the NewWindow area—there is a catch. I'm not going to promise that these routines will hold up after another revision of Intuition and Workbench, but hopefully they will need only to have their offsets corrected. The ROM Kernel Manuals make a point of using InitRequester to clear Requester structures for use, perhaps on the chance that, in the future, the default values for some of the fields might be values other than zero. By using InitRequester, you allow the MKREQ routine a slightly higher chance of not needing major revision after changes to Intuition. This makes it worth following the RKM, and is a heck of a lot faster than a FOR...NEXT loop, anyway.


```
tx% = ( w% / 2 ) - (( LEN( t$ ) * 8 ) / 2 )
MKITEXT tx%, 6, 2, 0, 0, t$, itp&, its&
```

These lines pass on the correct settings for the requester's IntuiText structure. The tx% variable is calculated so that the message is centered at the top of the requester.

```
POKEW rp& +  4, 0
through
POKE  rp& + 30, 1
```

This bunch of POKEs sets up the Requester structure in memory. Note that you set the values of the TopEdge and LeftEdge to zero. This is because you want to locate the requester based on the window location, not the requester itself.

```
x1% = 5
x2% = w% - 5
y1% = 3
y2% = h% - 3
POKEW rxy&,     x1% : POKEW rxy& + 2, y1%
through
POKEW rxy& +16, x1% : POKEW rxy& +18, y1%
```

As they do in the MKBTN routine, these lines set up the XY data for the Border structure—only this time for the requester. The X and Y values are calculated so that the border graphic is inside the outer edge of the requester boundaries.

```
POKEW rbp&,      -1
through
POKEL rbp& +12, 0&
```

In case you couldn't guess, here the routine sets up the Border structure linked to the Requester structure. For a slightly different appearance, try changing the POKE that sets the FrontPen to 3 instead of 2. This will create an orange border around the requester box, breaking up the monotony of all the black-on-white graphics.

```
FOR i& = wp& TO (wp& + 47)
  POKE i&, 0
NEXT i&
w% = w% + 8
h% = h% + 12
```

The FOR...NEXT loop here clears out the NewWindow area of the allocated memory—albeit slowly. Corrected Width and Height fields are then calculated, to adjust for the edge graphics of the window. Making w% and h% increase by more than 8 or 12 will result in a larger blue border around the requester graphic.

```
POKEW wp&,      x%
through
POKEW wp& +46, 1
```

Surprise, surprise. Here you set the values for the fields you will use in the NewWindow structure. We don't set all the fields, and the ones that remain zero Intutuion knows how to deal with. Setting Flags to 5168& gives you the ACTIVATE | SIMPLE_REFRESH | WINDOWDRAG | GIMMEZEROZERO setting you want. The value is found by OR'ing the values for all of the settings together.

```
x% = its&
END SUB
```

The MKREQ routine ends by setting x% to return the length of the IntuiText structure (all the other return variables are set) and handing the execution back to the main program.

```
mp&(4) = rp&
msiz%(4) = rx%
```

The main program then puts the pointer to the requester memory block and the size of its IntuiText into the arrays you are using for memory accounting. Now, everything in memory is set up to define a proper Requester unit.

```
CALL DoReqst&( wp&, rp&, dloc&, SADD(nam$) )
resp% = PEEKW( dloc& )
PRINT
PRINT "You selected choice number";resp%;"!!"
```

You now call the assembly routine using the pointer you created to point at the block of memory where LOADSBR put the routine. You pass the routine a pointer to the NewWindow structure, a pointer to the Requester structure, the address of the memory where you want the result placed, and the address of the "intuition.library"+CHR$(0) data. Because of BASIC's disconcerting habit of moving string variables around in memory, you use SADD at the routine instead of setting a variable. You want to make sure the routine gets where the data is now located, not where it was before BASIC moved it somewhere.

When the assembly routine returns, you pull the selected button's GadgetID value from the data memory where you told DoReqst to put the result. You can tell that DoReqst malfunctioned if the value returned is zero, and none of the GadgetID's were set to zero (generally a good practice). This means that the assembly routine had a problem with the data you handed it, and had to abort. More graphic examples of signs that you really messed up the data are system lock-ups and visits from Mr. Guru. After pulling the result from memory, a quick message is printed out which indicates the selection made from the Requester.

The following collection of lines in the FOR...NEXT loop take the address of each button's memory block and extract the address of the related IntuiText structure. That data is used to call MEMFREE to de-allocate all the memory allocated for the button structures and data. The same is then done for the memory used by the requester, NewWindow, and their related structures. The last MEMFREE call de-allocates the memory used by the assembly routine itself, effectively erasing it from memory. Do not attempt to call DoReqst after the memory holding the routine has been freed, or the consequences for your program could be disastrous!

```
LIBRARY CLOSE
END
```

At the end of the program, you close the libraries you opened and stop execution. In the words of Porky Pig, "Th-th-th-th-that's all folks!"

That's all there is to it. Once you become familiar with the routines, you will find them easily transported to your own programs. Setting up the requester unit is a little complicated, but the end result is requesters that work much more smoothly than ones written in BASIC.

As usual, feel free to hack and slash the routines as you see fit for use in your programs. Just remember the areas to be careful not to alter in the system structures (mentioned above). Unless you have enough experience to understand what the assembly routine is doing, don't change it, either. Altering the data for the routine without knowing exactly what you're changing is a quick and easy way to crash the system when the program is executed.

There are many possible modifications which could make the routines more useful. Perhaps by generating the data separately, and then loading it in as a block, you could keep many requesters in memory at once, simply by changing the pointers passed to DoReqst as needed. The routine will accept any valid LeftEdge unit, and can return the GadgetID of anything that makes a GADGETDOWN IntuiMessage.

Hopefully, these articles have made your BASIC programs better-looking and easier to use. Remember, with compilers giving the necessary speed, BASIC is coming back as a serious language for many application developers. With judicious use of system routines—along with a little assembly boost here and there—you can really make BASIC programs as serious as the rest.

Listing One: rBas

```
CLS
DECLARE FUNCTION AllocMem&() LIBRARY
LIBRARY "exec.library"
LIBRARY "intuition.library"
```

```
DIM mp&(4), msiz%(4)

DoReqst& = 0&
nam$ = "intuition.library"+CHR$(0)
dloc& = 0&

LOADSBR DoReqst&, dloc&

FOR i% = 1 TO 3
  b% = i%
  by% = 1 + ( i% * 15 )
  bt$ = "Choice" + STR$(i%)
  MKBTN 10, by%, 240, 12, 2, 5, 2, bt$, b%, bp&
  mp&( i% ) = bp&
  msiz%( i% ) = b%
  IF ( i% > 1 ) THEN
    bp1& = mp&( i%-1 )
    LINKBTN bp1&, bp&
  END IF
NEXT i%


rx% = 30
ry% = 30
bptr& = mp&(1)
rt$ = "Make Your Selection:"
MKREQ rx%, ry%, 260, 64, bptr&, rt$, rp&, wp&
mp&(4) = rp&
msiz%(4) = rx%

CALL DoReqst&( wp&, rp&, dloc&, SADD(nam$) )
resp% = PEEKW( dloc& )
PRINT
PRINT "You selected choice number";resp%;"!!"

FOR i% = 1 TO 3
  bptr& = mp&( i% )
  ts& = msiz%( i% )
  tp& = PEEKL( bptr& + 26 )
  MEMFREE bptr&, 80&
  MEMFREE tp&, ts&
NEXT i%

rp& = mp&(4)
ts& = msiz%(4)
tp& = PEEKL( rp& + 24 )
MEMFREE rp&, 196&
MEMFREE tp&, qs&

MEMFREE DoReqst&, 162&

LIBRARY CLOSE

END

SUB MKREQ ( x%, y%, w%, h%, bp&, t$, rp&, wp& ) STATIC

  MEMALLOCATE rp&, 196&
  CALL InitRequester&( rp& )
  rxy& = rp& + 112
  rbp& = rxy& + 20
  wp& = rp& + 148
  tx% = ( w% / 2 ) - (( LEN( t$ ) * 8 ) / 2 )
  MKITEXT tx%, 6, 2, 0, 0, t$, itp&, its&

  POKEW rp& +  4, 0
  POKEW rp& +  6, 0
  POKEW rp& +  8, w%
  POKEW rp& + 10, h%
  POKEL rp& + 16, bp&
  POKEL rp& + 20, rbp&
  POKEL rp& + 24, itp&
  POKE  rp& + 30, 1

  x1% = 5
  x2% = w% - 5
  y1% = 3
  y2% = h% - 3
  POKEW rxy&,     x1% : POKEW rxy& + 2, y1%
  POKEW rxy& + 4, x2% : POKEW rxy& + 6, y1%
  POKEW rxy& + 8, x2% : POKEW rxy& +10, y2%
  POKEW rxy& +12, x1% : POKEW rxy& +14, y2%
  POKEW rxy& +16, x1% : POKEW rxy& +18, y1%

  POKEW rbp&,     -1
  POKEW rbp& + 2, -1
  POKE  rbp& + 4, 2
  POKE  rbp& + 5, 0
  POKE  rbp& + 6, 0
  POKE  rbp& + 7, 5
  POKEL rbp& + 8, rxy&
  POKEL rbp& +12, 0&

  FOR i& = wp& TO (wp& + 47)
    POKE i&, 0
  NEXT i&
  w% = w% + 8
  h% = h% + 12

  POKEW wp&,     x%
  POKEW wp& + 2, y%
  POKEW wp& + 4, w%
  POKEW wp& + 6, h%
  POKE  wp& + 9, 1
  POKEL wp& +10, 32&
  POKEL wp& +14, 5186&
  POKEW wp& +38, w%
  POKEW wp& +40, h%
  POKEW wp& +42, w%
  POKEW wp& +44, h%
  POKEW wp& +46, 1

  x% = its&

END SUB

SUB MKBTN (x%,y%,w%,h%, c%, tx%,ty%,t$, id%, bp& ) STATIC
```

```
MEMALLOCATE bp&, 80&
xy& = bp& + 44
bdp& = xy& + 20
MKITEXT tx%, ty%, 2, 0, 0, t$, itp&, its&

POKEL bp&,      0&
POKEW bp& +  4, x%
POKEW bp& +  6, y%
POKEW bp& +  8, w%
POKEW bp& + 10, h%
POKEW bp& + 12, 0
POKEW bp& + 14, 3
POKEW bp& + 16, 4097
POKEL bp& + 18, bdp&
POKEL bp& + 22, 0&
POKEL bp& + 26, itp&
POKEL bp& + 30, 0&
POKEL bp& + 34, 0&
POKEW bp& + 38, id%
POKEL bp& + 40, 0&

x1% = 0
x2% = w%
y1% = 0
y2% = h%
POKEW xy&,     x1% : POKEW xy& + 2, y1%
POKEW xy& + 4, x2% : POKEW xy& + 6, y1%
POKEW xy& + 8, x2% : POKEW xy& +10, y2%
POKEW xy& +12, x1% : POKEW xy& +14, y2%
POKEW xy& +16, x1% : POKEW xy& +18, y1%

POKEW bdp&,     -1
POKEW bdp& + 2, -1
POKE  bdp& + 4, c%
POKE  bdp& + 5, 0
POKE  bdp& + 6, 0
POKE  bdp& + 7, 5
POKEL bdp& + 8, xy&
POKEL bdp& +12, 0&
    id% = its&

END SUB

SUB LINKBTN ( bp1&, bp2& ) STATIC

  POKEL bp1&, bp2&

END SUB

SUB MEMALLOCATE ( ptr&, siz& ) STATIC

  ptr& = 0&
  ptr& = AllocMem&( siz&, 65537& )
  IF ( ptr& = 0& ) THEN
    PRINT "Memory allocation error - size ";siz&
    STOP
  END IF

END SUB

SUB MEMFREE ( ptr&, siz& ) STATIC

  CALL FreeMem&( ptr&, siz& )

END SUB

SUB MKITEXT ( x%, y%, fc%, bc%, m%, t$, itp&, its& ) STATIC

  t$ = t$ + CHR$(0)
  its& = 20 + LEN( t$ )
  tl& = LEN( t$ )
  MEMALLOCATE itp&, its&
  tp& = itp& + 20

  POKE  itp&,     fc%
  POKE  itp& + 1, bc%
  POKE  itp& + 2, m%
  POKEW itp& + 4, x%
  POKEW itp& + 6, y%
  POKEL itp& + 8, 0&
  POKEL itp& +12, tp&
  POKEL itp& +16, 0&

  CALL CopyMem&( SADD( t$ ), tp&, tl& )

END SUB

SUB LOADSBR ( sptr&, dptr& ) STATIC

  MEMALLOCATE sptr&, 162&
  dptr& = sptr& + 160
  RESTORE MLData

  FOR i& = sptr& TO (sptr& + 158) STEP 2
    READ op%
    POKEW i&, op%
  NEXT i&

  POKEW dptr&, 0

MLData:
DATA &h48e7, &hfffe, &h2c78, &h0004, &h701d
DATA &h226f, &h004c, &h4eae, &hfdd8, &h2a40
DATA &h6700, &h0084, &h286f, &h0048, &h282f
DATA &h0044, &h206f, &h0040, &hcb4e, &h4eae
DATA &hff34, &h2a00, &h6766, &h2245, &h2044
DATA &h4eae, &hff10, &h6754, &hcb4e, &h2045
DATA &h2668, &h0056, &h102b, &h000f, &h7e00
DATA &h01c7, &h204b, &h4eae, &hfe8c, &h2200
DATA &h660c, &h3014, &h662c, &h2007, &h4eae
DATA &hfec2, &h60ea, &h2240, &h2429, &h0014
DATA &h2469, &h001c, &h4eae, &hfe86, &h0882
DATA &h0005, &h2002, &h6702, &h60d2, &h3014
DATA &h66ce, &h302a, &h0026, &h3880, &h60c6
DATA &h2245, &h2044, &hcb4e, &h4eae, &hff88
DATA &h2045, &h4eae, &hffb8, &hcb4e, &h224d
DATA &h4eae, &hfe62, &h4cdf, &h7fff, &h4e75

END SUB
```

•AC•

This Fall see all the best of the Amiga

AC's GUIDE To The Commodore AMIGA®

When the nights turn crisp and your thoughts return to your Amiga, let AC's Guide To The Commodore Amiga be your guide.

A Harvest of over 2100 products by more than 400 vendors!

During the summer, Amiga developers throughout the world have been diligently preparing Amiga products for all your needs. Now showcased in this, the largest Amiga-specific publication ever, is the harvest of their labors. Your guide will be an indispensable tool to doing more with the Amiga. Don't miss it!

Available at your local Amazing Dealer

C Notes from the C Group

Writing utility programs

by Stephen Kemp

This month we will discuss writing utility programs. A utility program is any program that performs a simple, useful task. For instance, last month's column included a utility program that would change all the characters in a file to upper or lower case depending on what was indicated on the command line.

Most utility programs follow a similar syntax. The command line is used to indicate input files, commands, and control switches. Occasionally, it may be necessary to ask for additional information after the program has been invoked. This month's example will include such a feature.

This program is called LOOK (see Listing One). It will search through all indicated files, looking for a search string that is entered after the program begins. If the skeleton program, GENERIC.C (included in C Notes, V4.8), or even CASE.C (V4.9) is available, make a copy named LOOK.C. Remember, when it comes to utility programs, it is usually easier to begin with an existing shell. If GENERIC.C and CASE.C are unavailable, the entire source of LOOK is provided in Listing One.

The specifications for this program are these:

1) File names, with or without wildcard characters, are input via the command line.

2) The search criteria should accept any string up to a prescribed length.

3) The user should have the option of finding an exact match for the search criteria (case-sensitive) or a match with any string that has the same letters, whether capital or not (case-insensitive).

4) Each line that matches the search criteria should be displayed for the user.

These tasks are fairly simple to program using many of the functions found in the C library. I have the MANX compiler and have used functions from its library. If another compiler is being used, find a corresponding function (or a combination of functions) to perform these tasks. This program has two functions that were not available in my standard library.

The program first checks to determine whether a sufficient number of parameters has been received. Remember, the first argument (argv[0]) will be the name used to invoke the program, so it can be skipped. The program then looks at all the arguments to determine whether any switches supported by the program are indicated. You should use a switch indicator character, like this program does using the hyphen, to help distinguish the flags. LOOK accepts a -C from the command line to indicate that the search should be case-insensitive. This means that a word like DOS would match dos, Dos, DOS, or any word containing the letters d-o-s consecutively. If -C is not indicated, an exact match of the letters and case must be found.

The command line is only used to enter the search files and the case switch if required. Because the search string can be just that—a string—it is easier for the program to ask for it, rather than try to retrieve it from a command line. This is because, to enter a string on the command line, it must be enclosed in quotes. An example of how this might look follows:

```
LOOK *.c "if (str.c == 10)"
```

If quotation marks are not used, the program will not be able to distinguish the "words" from file names. Suppose this command line was accidentally entered:

```
LOOK *.c if (str.c == 10)
```

Without quotes, the program would assume that each space-delimited item was a different argument. This program is innocent enough because it does not change anything in a file, but the hazard is obviously there. Also, even if entered via the command line using quotes, special editing would have to be included to determine whether or not the search string is supposed to contain the quotes. A simple way to avoid these problems is by asking for the additional input at the beginning of the program.

A simple prompt, issued by printf, is used to indicate when to begin entering the search string. The search criteria is accepted by the function fgets. Fgets was used because it accepts a maximum length as one of the parameters. LOOK is limited, in this example, to search 30 characters. This is an arbitrary number and represents the maximum number of characters that I can type without making a mistake. Any size search buffer may be used, but be sure the character buffer is at least one character larger than the request. This leaves room for the NULL at the end of the string. The "sizeof" directive was used here to ensure that the input size was appropriate. (Actually, fgets will stop at "max-1" character automatically, but I consider it good coding practice to always make string buffers 1 larger than required.)

After retrieving the user input, the newline character must be removed from the buffer, since this function adds it when the user presses ENTER. Some library input functions do not include the newline character, so it is always a good idea to check the documentation to decide if it must be handled. Finally, if a -C was included (now indicated by the variable "flag"), the function strupr will be used to change the search string to all upper case. This is done to simplify finding the string. A case-insensitive search requires that everything be "reduced" to a common denominator. LOOK uses upper case as the common denominator.

Now each file that matches the criteria is opened and searched. Each line is read, copied into an alternate buffer, then searched for the specified string. Since the line will be changed to upper case if the flag indicates a case-insensitive search, the alternate buffer is the one that is outputted if the search is positive. When the entire file has been read, only those lines containing the search criteria will have been printed. The file is closed before beginning again.

The search function requires a little more discussion. The method that strsrch uses to find the search criteria is not unusual. In fact, it is the brute force method. Although it suffices for this example, you may be interested in discovering faster, more flexible searching techniques. Entire books have been written on the subject of string searching and pattern matching. Also, greater flexibility can be provided by adapting a "regular expression" technique. However, these topics are not usually considered beginner material.

The input line is searched for the first character in the criteria. If this character is found, a pointer is returned with its position. A simple compare is then performed for the length of the search string. If an equal match is found (indicated by a 0 return value), then TRUE is returned from this function. But if the strings do not match, two very important things occur. First, the input string pointer is checked to see if it is pointing at the NULL character. If it is at the end of the string, there is no point in continuing. If not, then the for-loop cycles and the string pointer (str) is incremented (str++) past the character position that is currently held. THIS IS VERY IMPORTANT! If the pointer is not incremented, the program will get stuck in an endless loop, continually finding the same character position in the string again and again. Likewise, it is possible to increment past the NULL and begin searching random memory if you are not at the end of the string.

That's about it. I'll leave the rest of the code for personal examination. It is fairly straightforward and, with the exception of the string searching function, resembles the program we did last month. After trying out the program, go back and do a few experiments. Try adding a line counter to identify the number of those lines with matching criteria. Then try making a new switch that will stop searching the file once the first match is found—make whatever revisions you choose to help you get the most from your program.

If you have any questions or comments, you can write to me c/o Amazing Computing, P.O. Box 869, Fall River, MA 02720.

Listing One

```
/* LOOK.C is a program that will search through the specified    */
/* files for a given search string.                             */
/* The parameters for LOOK can be file names with/without       */
/* wildcard characters (using * and ?).                         */
/* An optional -C switch is also supported to indicate that     */
/* the search should be Case Insensitive.  Otherwise the        */
/* search criteria must match the string specified.             */
/* The user will be prompted to enter the Search Criteria       */
/* (string) once the program has begun.                         */
/* This program was written for the MANX C compiler             */

#include "stdio.h"
#include "fcntl.h"

extern char   *scdir();              /* directory function*/
extern short  access();              /* file accessible function*/
extern char   *index();              /* string index search function*/
#define TRUE  1
#define FALSE 0

main(argc,argv)                      /* program start */
short argc;                          /* argument counter */
char *argv[];                        /* argument variable pointer(s)*/
{
    char *fptr;                      /* pointer to a filename*/
    short cnt;                       /* counter for files    */
    short ndx;                       /* index for arguments  */
    char  flag;                      /* case flag            */
    FILE  *ifp;                      /* file io pointer      */
    char  srch[31];                  /* search string        */
    char  lbuf[256],altbuf[256];     /* line buffers         */

    if (argc <= 1){                  /* if not enough arguments provided
*/
                                     /* exit point if error discovered */

        printf("Search a file for a given string \n"); /* announce */
        printf("LOOK -C [file pattern] ...\n");        /* example */
        printf("  where -C is a Case Insensitive search\n");  /*flag */
        printf("  pattern may contain wildcards (* and ?)\n"); /*
lower*/
        exit(0);                     /* exit the program */
    }

/*———— Check for flags and switches here ————*/

    flag = FALSE;                                    /* indicate no flag*/

    for(ndx = 1; ndx < argc ;ndx++){                 /* examine arguments*/
        if (argv[ndx][0] == '-')                     /* found an
indicator*/
            if (argv[ndx][1] == 'c' || argv[ndx][1] == 'C')
                flag = TRUE;                         /* store the flag */
    }
    if (flag)                        /* if case insensitive search */
        printf("Case Insensitive search\n"); /* Send message */

/*————————————————————————————*/

    printf("Enter Search String:");      /* ask for string */
    fgets(srch,sizeof(srch)-1,stdin);    /* get the response */
    srch[strlen(srch)-1] = '\0';         /* remove newline character*/
    if (flag)                            /* if no case */
        strupr(srch);                    /* use upper case for compare*/
    printf("%s\n",srch);                 /* send newline */

/*————————————————————————————*/
```

```
    for(ndx = 1; ndx < argc ; ndx++){                   /* examine arguments*/

        if (argv[ndx][0] == '-')                        /* flags already
found*/
            continue;                                   /* so skip to next */

        for(cnt=0;(fptr=scdir(argv[ndx]));cnt++){/* look for wildcards
*/
            if (access(fptr,0) == 0){                   /* if file is found
*/
                printf("*** %s ***\n",fptr);            /* display filename
*/
/*------  Utility program specific code goes here ----------*/

                ifp = fopen(fptr,"r");                  /* open file for read*/
                if (ifp == NULL){                       /* if opens fail*/
                    printf(" - skipped - \n");
                    continue;                           /* continue looping*/
                }
                for(;;) {                               /* loop through file
*/
                    if (fgets(lbuf,256,ifp) == NULL) { /* EOF */
                        fclose(ifp);                    /* close file */
                        break;                          /* end loop */
                    }
                    strcpy(altbuf,lbuf);                /* copy the line */
                    if (flag)                           /* if flag */
                        strupr(lbuf);                   /* up the string */

                    if (strsrch(lbuf,srch) == TRUE)  /* if Found */
                        printf(altbuf);                 /* print original
line*/
                }

/*-------------------------------------*/
            }else{                                      /* if not found */
                printf("Cannot find %s\n",fptr);/* indicate */
            }
        }
        if (cnt == 0)                                   /* no matches */
            printf("Cannot find %s\n",argv[ndx]);/* for this argv */
    }
}

/* This function searches a given string (line) for the string (srch)*/
/* and returns TRUE if found and FALSE otherwise.  */

short strsrch(line,srch)
char *line, *srch;
{
    char *str;
    short len;

    len = strlen(srch);                                 /* determine length*/

    for( str = line; ; str++){                          /* loop through string */
        str = index(str, *srch);                        /* look for this
character*/
        if (str == NULL)                                /* if not found */
            break;                                      /* terminate loop */
        if (strncmp(str,srch,len) == 0)                 /* Found it! */
            return(TRUE);                               /* return TRUE */

        if (*str == '\0')                               /* if at end of string*/
            break;
    }
    return(FALSE);                                      /* return FALSE */
}

/* strupr is a function that indexes through a string and converts */
/* all alphabetic characters to upper case */
strupr( str )
char *str;
{
    for( ;*str != '\0'; str++)     /* search until null is found */
        *str = toupper(*str);      /* call upper case function */
}


[------End of listing one--------]
```

•AC•

Worth The Wait:

JForth Professional 2.0

review by Jack Woehr

It is only fitting that the microcomputer world's most brilliant hardware creation, the Amiga, should be garlanded with the most powerful software systems available. JForth Professional from Delta Research of San Rafael, California fulfills this requirement from the perspective of the professional Forth programmer.

JForth Professional is a complete software development system centered around a state-of-the-art, full 32-bit JSR-threaded Forth. Forth programmers have always been able to point to the convenience of programming in Forth, but rarely have they been able to point so confidently to the execution speed of a Forth system as they can now with JForth. When JForth 1.0 was introduced in 1986, there was no comparable Forth system on any popular microcomputer. JForth set the standard, and the Forth world raced to catch up.

JForth provides an excellent environment for Amiga programmers, from the rankest amateur to the professional. Forth is a compiled language with an interpretive layer. Code may be added to the system simply by creating a "colon definition"—typing in the name of the routine preceded by a colon, then listing the "older" routines which the new routine is to call, and terminating the definition with a semicolon. Such definitions may be added to the system at any time, either by typing them in directly or by compiling a text file.

In traditional Forth systems, definitions are compiled into lists of addresses of the called routines. These in turn may consist of address lists and so on until, at some lowest level, executable machine code is reached. Such definitions must be "interpreted" by an "inner interpreter", a set of routines, themselves written in Forth, which understands the address lists.

This is not the case with JForth. Short code routines, when referenced in a definition, are compiled as inline code. (The maximum length of words to be compiled inline is a programmer-settable option.) References to previously defined routines—which are called rather than compiled inline—are laid down as optimized JSR's or BSR's in a clever register-addressed scheme that avoids the use of token tables.

The entire body of a JForth JSR-threaded definition is executable MC68k code. Every definition ends in an RTS. This is the fastest executing type of Forth which can be implemented on a traditional microprocessor. For a faster micro Forth you would have to buy a Harris RTX2000, a chip designed by Forth author Charles Moore specifically to execute Forth in silicon.

While JForth enjoys compiler language execution speed, the programmer never needs to sacrifice the convenience of having a fully interpretive language at his disposal. One obvious advantage of such a system is that you do not have to learn about making Amiga library calls by constant compiling, loading, and crashing. You can open windows, create graphic objects, and move things around windows while controlling them interpretively—without sacrificing compiled-code speed. You can even crash the system interpretively, a great time saver over traditional compilers!

JForth comes on two disks. The first disk, JForth:, includes most of the source code in Forth for the entire system, many utilities, and the Amiga Include and FD files. The second disk, the Extras: disk, contains the minimal kernel of JForth (created from Assembly), for which the source code is not included, and two other JForth images produced by having compiled various assortments of the extensions and utilities available in source on the two disks.

The recommended precompiled development image of JForth is about 161K. The Extras: disk also includes the system-generating code for expanding the kernel to a full development system in the Sysgen drawer. The Demos drawer contains many instructive and attractive demos with full source. The Appls drawer includes several intriguing and useful applications (including a wordcounter, a prettyprinter, and a very dumb terminal program). The Clone drawer contains the complete source to CLONE. More on CLONE in a moment.

A session with JForth Professional starts by clicking the XICON scripts on the distribution disks. The scripts assign logical device names to several of the JForth drawers. JForth has a full interface to the normal Amiga file system, and can call up the file from which a word in the JForth image was compiled and display the file for the programmer. JForth "knows" the source file locations under these logical device names.

"JForth provides an excellent environment for Amiga programmers, from the rankest amateur to the professional."

Double clicking on another icon brings up JForth. System initialization includes setting up traps in the 680x0 vector table so that programmer errors which should incarnate the Guru, such as odd address errors, instead evoke a mild JForth rebuke text. Of course, if the programmer is masochistic enough, he or she can compile the file RUDE.F, which generates truly shocking red-and-black system alerts. Dictionary hashing is also installed during JForth system initialization. The JForth Professional dictionary is

hashed for faster searches during compiles. Compiles are now quite a bit faster than in earlier versions of JForth.

At any time the programmer can execute MAP, which displays a map of the system as currently constituted. A new image with a larger dictionary space can be created by changing a system variable and executing SAVE-FORTH.

For those familiar with more common Forth systems, there are quite a few surprises in JForth, most of them quite pleasant. As mentioned above, JForth definition bodies consist entirely of executable code. A quick DUMP of part of the dictionary is very informative on this point. Typing DEF or SEE <wordname> results in disassembly, rather than decompilation. However, if branches of the disassembled Forth word point to other high-level words possessing a name header, the name of the word branched to is printed in parentheses next to the disassembled branch.

JForth Professional CLONE is an optimizing target compiler. This means that once you have perfected your program under JForth's normal interpreter/compiler environment and you are satisfied with your code, you can invoke CLONE and produce from the compiled program an optimized, minimally sized target image suitable for commercial distribution. The output of CLONE is not a Forth system; it is your application program pared to the bone.

JForth Professional comes with the complete source to ODE, a congenial and complete object-oriented extension to JForth which allows the programmer to explore Oops! programming style without losing access to the underlying simplicity of Forth syntax. ODE has early and late binding, classes, methods and inheritance—all the tools the OOPS programmer has come to expect.

Among the 91 utilities in the JForth:util drawer are: floating point math, local variables, random numbers, graphics code, printer logging, multi-standard package (a useful item if you intend to port code over from other Forth systems), BLOCK support (JForth uses regular Amiga text files as program source; BLOCK is a loadable option), and a JForth implementation of the routines contained in amiga.lib mentioned in the Amiga manuals.

JForth Professional features detachable precompiled modules, so you can use certain JForth resources without adding them permanently to the Forth dictionary. Among the modules are a set of Amiga Includes, an assembler and the disassembler. Source code for the modules and for the code to create new modules from your own code is included on the JForth distribution disks.

Incidentally, the disassembler is not limited to the JForth image. Nor is the power of words like ! (store) DUMP and @ (fetch) limited to the JForth image. The JForth programmer can wander around the memory like a worm, "peeking and poking" at will. (Watch out for collisions with system structures!)

JForth actually has two assemblers, one "traditional" reverse-polish Forth-style assembler, and one featuring Motorola forward syntax with local labels. Use the one which suits your needs. The Motorola syntax assembler is especially useful in writing code words or stand-alone routines such as interrupt handlers. The advantage of the reverse-polish Forth-style assembler is that one can resort to assembler right in the middle of a high-level Forth definition without any "magic handwaves".

Another gratifying feature of the JForth interpretive/compiler environment is that JForth has F-key mapping and command-line history and editing like that provided by the shell. The F-keys come premapped to such useful words as INCLUDE and MAP, but this can be changed at any time by the programmer. As usual, the source for these items which comes precompiled in the recommended development image is included on the distribution disks.

JForth Professional comes with a laser-printed manual of about 300 pages in a three-ring binder. The various features of the system are described in detail. There is a tutorial for beginners in Forth, and a glossary of the JForth vocabulary. ODE, Amiga system calls, and the JForth version of C structures are all carefully documented.

JForth is the brainchild of Phil Burk, Brian Donovan, Mike Haas and Jim King. Phil and Mike recently used JForth and an Amiga to win $1,000.00 and the title of World's Fastest Programmers at the Realtime Programming Convention last November in Anaheim, CA.

The World's Fastest Programmer Contest required entrants to bring their own hardware and software and use it to program a "Mystery Gizmo" controller to wave a sawblade around in the air while a set of LED's flashed out "THE RAIN IN SPAIN FALLS MAINLY IN THE PLAIN". With JForth and the Amiga as their development platform, Phil and Mike had the one grand in the bag in a little over an hour, while other programmers were fuming, fussing, and burning out the delicate stepper motors of the Mystery Gizmo.

JForth Professional is alternatively known as JForth 2.0, the culmination of three years of debugging, improvements, extensions, and optimization of the original JForth 1.0. Thereby hangs the tale.

In 1986 when JForth 1.0 was first announced, the ads promised, among other things, "Optimizing Target Compiler" and free upgrades for the cost of media and postage and handling. Early purchasers eventually received the upgrade to version 1.2, which up to now has been the latest commercial release of JForth. However, the optimizer was not forthcoming, so to speak. In its place was a Turnkey utility that stripped the dictionary headers from the system, thus manufacturing a legally distributable object that did not constitute a redistribution of the underlying JForth system.

However, the object was not automatically optimized, and the programmer had to very carefully compile a minimal image by recompiling the system upwards from the kernel to achieve a distributable object of reasonable size. Furthermore, there was no way to shrink the kernel itself, meaning that any JForth program had a minimum size of about 70K.

CLONE, however, is the promised goods, and does indeed produce objects as small as about 2 1/2K. However, much has changed with JForth other than CLONE, and Delta Research came to market with the greatly expanded JForth Professional system, JForth 2.0, offering $50.00 upgrades to registered 1.2 owners.

At this point, some of the earliest buyers who had purchased on the basis of the original ad (which had been long cancelled even before version 1.2 was available) contacted Phil Burk of Delta Research and asked why Delta should not keep its original commitment to a target compiler. Phil responded by offering a CLONE-only upgrade to registered 1.2 owners for $10.00, which includes the cost of media, postage and handling. This reviewer, who has followed and owned JForth since version

1.0, is glad to see Delta honoring its early advertisements.

JForth has its quirks, some apparently rooted in the collective personality of the Delta team. While JForth is advertised as "Forth 83 Standard". This is an impossibility. The 1983 Standard (unwisely, in many opinions) defined Forth in terms of 16 bits. A 32-bit Forth cannot be 83-Standard, though it may resemble the standard quite closely from the programmer's point of view.

Further, philosophical differences between Delta and certain trends of the Forth 83 Standard led to some anomalies: NOT in JForth is a logical Boolean NOT instead of the bitwise NOT prescribed by the 83 Standard. Division is floored to zero (0) in JForth, instead of towards negative infinity, which is the 83 Standard. The familiar division operator UM/MOD is missing and is replaced by U/ which operates in a slightly different manner.

Delta Research has not been alone in its conflict with the 83 Standard. The world's largest Forth vendor, FORTH Inc., has never totally reconciled itself to the 83 Standard. The question may soon be moot. The Forth language is in ANSI X3J14 procedure right now, and if anything may be gleaned from the BASIS documents leaking out from the committee, the ANSI Standard Forth to emerge in 1990 or 1991 will contain radical departures from the 83 Standard, which later was very heavily tilted towards 16-bit micros.

In any event, the Forth standard is currently undergoing revision at the ANSI level to deal with the problems vendors faced as Forth grew beyond the horizons envisioned by the 1983 Standards committee. As a rooter on the sidelines of the current Standardization effort, it is my observation that JForth substantially conforms to the functionality of the still-evolving ANSI Standard BASIS.

Some of the observed anomalies in JForth are addressed by the JForth loadable multi-standard package which allows the programmer to go back all the way to FIG Forth if desired, but anyone intending on porting code from other Forths should be aware that the old saw, "If you've seen one Forth, you've seen...one Forth" is not ready for retirement quite yet.

Experienced Forth programmers will have mixed feelings about the absence of local multitasking in JForth.

Phil and Mike feel local multitasking is inefficient on the Amiga; they recommend using the Amiga Exec facilities for spawning child processes. Your correspondent favors the syntactic convenience of Forth-style local multitasking and hopes to prevail upon the Delta gang to hook up all the stubs they purposely left in the kernel for local multitasking. Those stubs include full implementation of local variables, and make it possible for the programmer to install local multitasking on his own! I may try it Real Soon Now.

However, not one of the quirks of JForth renders the system unusable or even difficult to use. Forth programmers just tend to be eccentric and opinionated (talk to me sometime), and the Delta gang is no exception!

Most important, I cannot envision a more suitable environment for programmers unfamiliar with the Amiga system resources to pleasantly and efficiently explore the intricacies of the Exec-Intuition-AmigaDOS universe. The value of the interpretive layer, combined with the compact syntax of Forth and the blazing execution speed of JForth, should not be taken lightly by Amigans nor professional Forth users.

•AC•

Author's Info

Jack Woehr is a Forth programmer at Vesta Technology, Inc. in Wheat Ridge, CO. He is Chapter Coordinator for the Forth Interest Group, and host of the Forth Conference on the WELL in Sausolito, CA. Mr. Woehr is also Sysop of the RealTime Control & Forth Board, which is part of the CFB Forth Network. Jack Woehr can be contacted through the GEnie network, or you can write to him c/o Amazing Computing, P.O. Box 869, Fall River, MA 02722-0869.

Delta Research
P.O. Box 1051
San Raphael, CA 94915
(415) 485-6867
JForth Professional 2.0, $179.95
Inquiry # 219

They slice, they dice,
they make using gadgets in Assembly a breeze!

Glatt's Gadgets

by Jeff Glatt

In my last article "Getting Started in Assembly" (AC V3.12), I presented a skeletal example program that used Amiga libraries, windows, and menus, and communicated with Intuition. In this article, I will add some gadgets to the window.

Gadgets are visual items inside windows and requesters which can be manipulated by the user to enter data. There are several types of gadgets. Some are system gadgets, and are primarily used by Intuition. The small box at the bottom right corner of some windows which lets you resize it with the mouse select button, is an example of a system gadget. So are the front and back depth-arrangers, the window-close box, and drag bars. You can add any of these gadgets to any window that you open by specifying certain flags in the newWindow's window flags field. For a list of window flags, see the "Libraries and Devices" section of the ROM Kernel Manual (page D-154, beginning at line 794).

There are also application gadgets. These are gadgets that you must define yourself. You need to set up various structures that define what each gadget does, what it looks like, and where it will be located. The three types of application gadgets are string, boolean, and proportional gadgets. String gadgets are used to enter text from the keyboard. They are commonly seen in file requesters, where they are used to type in specific filenames. Boolean gadgets have only one of two states: selected or unselected. Think of them as switches. An example is a gadget labeled "CANCEL" or "OK" seen in various DOS requesters. Proportional gadgets look like "sliders" or "faders." They have a knob that moves around inside a closed box. An example of this is the key-repeat slider in Preferences.

An example of a gadget structure is myPotGadget in Listing One. The first LONG in the structure is the address of the next gadget in the same window or requester. You create a linked list of gadgets by having that next gadget's NextGadget field point to a third gadget, etc. When you finally get to the last gadget in the window, its NextGadget field must be zero.

The next two WORDS of the structure describe where to place the gadget (relative to the top left corner of its window or requester, unless you specify GRELBOTTOM or GRELRIGHT flags, which represent how many screen pixels to move to the left and down, respectively). If you specify GRELBOTTOM or GRELRIGHT, then the two numbers are negative offsets from the the window's right and bottom borders. The advantage of using these is that if your window has a WINDOWSIZING system gadget, then your application gadgets will "move" with the window as it is resized. Otherwise, the gadgets might "disappear" as the window is made smaller.

The next two WORDS are the width and height, in screen pixels, of the gadget select box. Note that this is the size of a rectangular area in which the user can "click" to select the gadget. This does not determine the size or shape of the gadget's graphics. An IntuiImage or Border will do that.

If you skip the next two WORDS of the structure, you'll see a field labeled Type. This is where you describe what type of gadget you're dealing with. A list of gadget types can be found in the "Libraries and Devices" section of the RKM (page D-148, beginning at line 332). Since this is an application gadget—as opposed to a system gadget like the drag bar—you need to CLEAR bit 15 of this WORD. Also, this gadget is for a window—as opposed to a requester—so you should CLEAR bit 12, as well. You want the gadget attached to the window, and not to the screen, so bit 14 is CLEARed also. Since this gadget won't be placed in the window's border, you don't need GIMMEZEROZERO borders, so bit 13 should be CLEARed. Here are the bits to SET for each of the available types of application gadgets:

BOOLEAN GADGET	Bit 0
PROPORTIONAL	Bit 0 AND Bit 1
STRING GADGET	Bit 2

Since myPotGadget is intended to be a proportional gadget, I SET bits 0 and 1. In the gadget structure is a Flags field. A list of available flags, their hex values, and what they do can be found in the "Libraries and Devices" section of the RKM (page D-147, beginning at line 248).

I already mentioned GRELBOTTOM and GRELRIGHT. There are two other similar flags, GRELHEIGHT and GRELWIDTH. These change the gadget's height and width, respectively, as the window is resized. If your window doesn't have a resizing gadget, then these four flags will not be of any use to you.

Bit 7 of the Flags field is used by Intuition to indicate when a gadget is being selected. It is SET while selected, and CLEARed when un-selected. Your program can test this bit if it wants to know if a gadget is in selected state.

If you want a special "design" in the un-selected state, you must create the image data (using an Image editor), place the data in CHIP memory, initialize an IntuiImage structure, and store that structure's address in the GadgetRender field of the gadget. In myPotGadget, I supplied my own image for the knob. An IntuiImage structure called myPotImage has been initialized, and the actual data for the image is at the label PotImageData. This data must be placed in CHIP memory, but

the gadget and image structures do not. I will allocate CHIP memory when the program is run, and copy PotImageData to this CHIP memory block before opening the window.

A good place to add this allocation is in the open_libs routine from my last article's code. At label B3, add the following code:

```
B3    moveq       #36,d0   ;the # of bytes in PotImageData
      moveq       #3,d1    ;means "give me PUBLIC, CHIP memory"
                           ;_SysBase already in a6
      jsr         _LVOAllocMem(a6)
      move.l      d0,myPotImage+10 ;store the address of the
                                   ;block in our
                                   ;IntuiImage structure's
                                   ;ImageData field.
      beq.s       B10      ;If no memory, we'll have to exit
                           ;===Now copy PotImageData to this
                           ;CHIP mem
      lea         PotImageData,a0
      movea.l     d0,a1
      moveq       #36,d0
CPYI  move.b      (a0)+,(a1)+
      dbra        d0,CPYI
```

This is added right after the call to FindTask. _SysBase is still in a6, and your window hasn't yet been opened.

Note the attributes in d1 sent to AllocMem. Setting Bit 1 means "give me CHIP mem," and setting Bit 0 means "make it known to other tasks and interrupt code." There are other specifications that you can make. Setting Bit 16 will automatically give you a block that has been initialized to all zeroes. You could set bit 2 if you wanted FAST mem, but do not do this unless you are certain that your Amiga has expanded memory (more than 512K). If you don't set bits 1 or 2, Exec will try to get FAST mem first, and then CHIP mem if FAST is unavailable. If you set bit 2 (FAST), and your Amiga has only 512K, then the allocation will fail (i.e., return zero).

Note that I am storing the address of the returned block in myPotImage's ImageData field. The remaining fields in myPotImage describe the size of the image, the address of a NextImage structure linked to it (not needed here), and some variables that describe how the image is to be rendered. A short explanation of this can be found in the "Libraries and Devices" section of the RKM (paeg D-151, beginning at line 536). Also note that the GadgetRender field of myPotGadget contains the address of myPotImage's structure.

You'll also need to de-allocate the CHIP memory when you exit the program. Add these lines at label C0 (at the beginning of quit_all):

```
C0    movel       myPotImage+10,d0
      beq.s       C1
      movea.l     d0,a1
      moveq       #36,d0
      movea.l     _SysBase,a6
      jsr         _LVOFreeMem(a6)
```

If your assembler supports it, you can declare the data as CHIP memory right in the assembly listing. This way, you won't have to allocate CHIP memory and copy the data into it. You do this by placing an assembler directive before the image data, like this:

```
SECTION imagedata,DATA,CHIP

PotImageData  ;put your data here
```

The SECTION statement is not a 68000 instruction. It is a special command that your assembler carries out when it assembles the code. After the CHIP data, you should put another SECTION statement without the CHIP specification.

```
SECTION notimage,DATA
```

Instead of an image, you can use a border to create the gadget graphics. In this case, the GadgetRender field must point to a Border structure, rather than to an IntuiImage structure. You'll also need to supply the data describing the points in the border, but it doesn't need to be in CHIP memory.

An example of a border structure is myStrBorder. The first two words give the X and Y coordinates of the border's origin. This position is relative to the gadget's top left corner. By specifying (-4,-3), I put the initial pen position 4 pixels to the left and 3 pixels up from the gadget's top left corner. The next 2 bytes are the Fore and Back pens. Remember, with a Workbench screen there are 4 choices (0 to 3). The Drawing mode is JAM1. The count field tells how many pairs of points there are in the border data. Each pair of WORDS of the data represents the X and Y coordinates of a point. The vector field is the address of the actual data points. For example, since the first two words of StrPts gives (0,0), the first point will be at the origin. Intuition draws lines between all the points in the array.

You tell Intuition whether the GadgetRender field is pointing to a Border or to an IntuiImage structure by CLEARing or SETting the GADGIMAGE bit (#2) of the gadget's Flags. If you leave the GadgetRender field 0, then no imagery will be displayed for the gadget. I SET the GADGIMAGE bit of myPotGadget's Flags, since I supplied an image, rather than a border. On the other hand, for the string gadget, myStrGadget, the GadgetRender field points to a Border structure, so myStrGadget has its GADGIMAGE bit CLEAR. (Note: For some reason, Intuition will not allow the use of a Border structure with only proportional gadgets. You can use an IntuiImage.)

The gadget structure has a SelectRender field that is used in conjunction with the following Flags to inform Intuition how to display the gadget when the user clicks on it:

```
GADGHCOMP       ($0000)  Complement the colors.
GADGHBOX        ($0001)  Draw a box around the gadget's
                         image.
GADGHIMAGE      ($0002)  Use the alternate image or border
                         (which you must supply).
GADGHNONE       ($0003)  Don't change the gadget in any way.
```

You need to choose one of the above. If you want GADGHIMAGE, then you must create the image or border data, along with its associated structure. If you used a border for GadgetRender, then you must use a border for SelectRender. The same is true for an IntuiImage. You store the address of this IntuiImage or Border structure in the gadget's SelectRender field. If you prefer GADGHBOX, GADGHCOMP, or GADGNONE, then set SelectRender to zero and don't bother with anything else. For these three, you don't need a border or image. GADGHCOMP changes the colors of the gadget when it is selected. GADGHBOX draws a box around the gadget, but this really works only for Boolean gadgets. Furthermore, string gadgets only allow GADGHCOMP or GADGHNONE as the highlighting mode. GADGHNONE doesn't change the gadget at all when it is selected.

Note that I have chosen GADGHCOMP as the highlighting method for myPotGadget. One of the fields in the gadget structure is for Activation flags. These determine what messages Intuition will send when the gadget is selected, in use, and unselected. Here is a list of the bits that can be set to enable certain features:

Name	Bit #	Function
RELVERIFY	0	Intuition sends a GADGETUP (release) if the mouse pointer was over the selected gadget when the user released the mouse button.
GADGIMMEDIATE	1	Intuition sends a GADGETDOWN (selected) as soon as the mouse is clicked on the gadget. (No MOUSEBUTTON is sent here.)
ENDGADGET	2	Tells Intuition to end (cancel) a requester or AbsMessage when gadget is selected.
FOLLOWMOUSE	3	Useful for proportional gadgets.Sends mouse move messages while the gadget is selected.
RIGHTBORDER LEFTBORDER TOPBORDER BOTTOMBORDER	4 5 6 7	If you placed the gadget "along" the border of the window (like Workbench's disk full indicator), set the bit of the border that you want adjusted to "encompass" the gadget.
TOGGLESELECT	8	Toggle the gadget's Select bit (#7) statein the flags field each time the gadget is selected.
STRINGCENTER STRINGRIGHT	9 10	String gadgets require you to supply a buffer into which Intuition can dump the user's keyboard input.These two indicate where in the buffer to place the cursor.
LONGINT	11	This string gadget ndicates that the user input is going to be an ASCII string of numbers which Intuition should convert to a LONG and place in the StringInfo's LongInt field.
ALTKEYMAP	12	This string gadget means that you have supplied an alternate keymap.

For myPotGadget, I want to know when the user selects it (GADGIMMEDIATE), I want to be sent mouse position messages (FOLLOWMOUSE), and I want to know when the user releases it (RELVERIFY). For proportional gadgets, the mouse doesn't have to be "over" the gadget when the button is released. So what Intuition is going to send me is the following (in this order):

1.) One GADGETDOWN message when myPotGadget is selected.
2.) Mouse move messages while the user moves the mouse around.
3.) One GADGETUP message when the user finally releases.

Note that in order for FOLLOWMOUSE to work correctly, my window's IDCMP flags must have set MOUSEMOVE, and the window Flags REPORTMOUSE.

There is a field in the gadget structure for GadgetText. This is the address of an IntuiText structure (with its TopEdge and LeftEdge relative to the gadget's top left corner). This is for you to add text to the gadget. I chose to add the word "POT" beneath the proportional gadget.

There is a field in the gadget structure labeled SpecialInfo. This field holds the address of a special structure. When you're using proportional gadgets, this must be a PropInfo structure. When you're using string gadgets, this must be a StringInfo structure. For boolean gadgets, this field should be zero.

An example of a PropInfo structure is myPotPropInfo. The first WORD in the structure describes the PotFlags. Here are the bits to set for various flags: AUTOKNOB (Bit 0), FREEHORIZ (Bit 1), FREEVERT (Bit 2), PROPBORDERLESS (Bit 3), and KNOBHIT (Bit 8).

If you SET AUTOKNOB, you do not have to allocate and initialize CHIP memory for the knob as I did with myPotGadget. You still need to have an IntuiImage structure (like myPotImage) and store its address in the gadget's GadgetRender field. But you needn't create any image data, copy it to chip memory, and store the address of the CHIP mem in the IntuiImage's ImageData field. Intuition will do this, thus supplying you with a default knob. (Also, you don't have to initialize the IntuiImage structure.)

FREEHORIZ means that the slider will be laid "on its side" and the knob will move across the screen left or right (along the Y-axis). FREEVERT means that the slider will be "stood upright" and the knob will move up and down (along the X-axis). You can specify both, and the knob will have 360-degree mobility. An example of this is the screen-position gadget in the middle of WorkBench's Preferences screen.

The next two WORDS are the HorizPot and VertPot fields. These two fields hold the current (X- and Y-, respectively) position of the knob within the slider. This will be a value from 0 to 65,535, with 0 being the position closest to the bottom of the screen for FREEVERT, or the position farthest to the left for FREEHORIZ. For myPotGadget, I have chosen FREEHORIZ (left and right movement only), so the HorizPot field contains the

information I need. I could use this value to determine the value of a certain variable in the program, or perhaps use it to determine the current position in a list. Many times, you won't need the full range of values from 0 to 65,535, but regardless of the dimensions of the gadget, you always get this range. You need to scale the value for your use. The last six fields (WORDS) of the PropInfo are for Intuition's use.

An example of a StringInfo structure is myStringInfo. The first LONG is the address of a buffer into which Intuition will place the user's input. The second LONG is the Undo buffer used by Intuition. This is optional (for no Undo, set it to NULL), and enables Intuition to restore the previous contents of the string gadget when the user presses [right Amiga]-Q.

The next WORD is the character position where the cursor should appear when selected. After that is the maximum number of characters (including the NULL terminator) which can be placed in the buffer. Make your buffer (and Undo) this length. The next WORD is the position within the buffer of the first character to display. The following variables are used mainly by Intuition. You may determine how many characters are in the buffer by examining the NumChars field. The LongInt field is used if you specified LONGINT in the gadget's flags. After the user types in an ASCII string of digits (with an optional leading sign), Intuition will convert the string into a SIGNED LONG and store it here. The last field is for supplying your own custom keymap if you have set the gadget's ALTKEYMAP flag.

Going back to the gadget structure, there is a field for MutualExclusion. This has not been implemented as of Workbench 1.2, so I suggest you ignore this until Commodore updates the OS. If you want to implement your own form of mutual exclusion, the Enhancer manual with 1.2 gives an example (in that awful language known as C). You'll have to turn it into real code via a non-real-time interpreter (commonly known as a compiler).

There is a field for the gadget ID. This can be any number (of your choice) from 0 to 65,535. You should pick a unique number for each gadget, and it is a good idea to reserve 0 for any CANCEL gadget. When receiving a GADGETDOWN IntuiMessage, you can extract the IAddress field (the base of the gadget structure that was selected), and get the gadget's ID in order to determine which gadget is in effect.

The last field in the gadget structure is for UserData. Since Intuition ignores this, you can use this LONG for any purpose you like.

In order to have your gadgets attached to the window, we need to store the address of the first gadget in the list (myPotGadget) in the newWindow's FirstGadget field before opening the window. Simply change the 0 in the original listing to myPotGadget. Adding or removing gadgets from an already opened window can be accomplished with Intuition's AddGadget and RemoveGadget.

Add the code in Listing One to the code in the last article, and you should see four gadgets when the window is opened: two proportionals (one with AUTOKNOB placed in the border of the window and a size relative to the window's), a String, and a boolean gadget. Of course, we still have not written a handler for the MOUSEMOVE, GADGETUP, and GADGETDOWN messages that Intuition is sending, but we'll do that next month. For now, experiment with positioning, sizing, and adding gadgets, as well as with changing imagery, borders, and IntuiText.

Assembler Listing

```
myPotGadget1:
  dc.l myPotGadget2 ;NextGadget (address of Next Gadget in
linked list)
  dc.w 125          ;LeftEdge
  dc.w 160          ;TopEdge
  dc.w 250          ;Width
  dc.w  10          ;Height
  dc.w  4           ;Flags = GADGHCOMP|GADGIMAGE
  dc.w $B           ;Activation = GADIMMEDIATE|FOLLOWMOUSE
                                  |RELVERIFY
  dc.w  3           ;Type = PROPortional
  dc.l myPotImage1 ;GadgetRender (address of this gadget's
                                  image structure).
  dc.l  0           ;SelectRender (If not NULL, supply an image
in CHIP mem)
  dc.l  PotTxt      ;GadgetText   (an IntuiText structure for
                                  this gadget)
  dc.l  0           ;MutualExclude
  dc.l myPotPropInfo1 ;SpecialInfo
  dc.w  0           ;ID (should be different for each gadget)
  dc.l  0           ;UserData, if any
myPotGadget2:
  dc.l myStrGadget
  dc.w  -15
  dc.w   10
  dc.w   15
  dc.w  -19
  dc.w  $50       ;Flags = GRELHEIGHT|GRELRIGHT|GADGCOMP
  dc.w  $B
  dc.w  3
  dc.l myPotImage2
  dc.l  0
  dc.l  0
```

```
  dc.l  0
  dc.l myPotPropInfo2
  dc.w  1
  dc.l  0
myStrGadget:
  dc.l  myBoolGadget
  dc.w 127
  dc.w 185
  dc.w 283
  dc.w  9
  dc.w  0          ;Flags = GADGHCOMP
  dc.w  1          ;Activation = RELVERIFY
  dc.w  4          ;Type = STRING
  dc.l  myStrBorder
  dc.l  0
  dc.l  0
  dc.l  0
  dc.l myStringInfo
  dc.w  2
  dc.l  0
myBoolGadget:
  dc.l  0        ;This field is 0 in the last gadget
  dc.w  20
  dc.w 160
  dc.w  61
  dc.w  11
  dc.w  0          ;Flags = GADGHCOMP
  dc.w  $101       ;Activation = RELVERIFY|TOGGLESELECT
  dc.w  1          ;Type = BOOL
  dc.l  BoolBorder
  dc.l  0
  dc.l  0
  dc.l  0
  dc.l  0
  dc.w  3
  dc.l  0
```

```
BoolPts:
  dc.w 60,10,60,0,0,0,0,10,61,10,61,1,62,1,62,11,1,11
BoolBorder:
   dc.w   0,0
   dc.b   2,0
   dc.b   0
   dc.b   9
   dc.l   BoolPts
   dc.l   0

PotTxt dc.b 1          ;FrontPen
       dc.b 0          ;BackPen
       dc.b 0          ;JAM1
       dc.b 0          ;Pad byte
       dc.w 3          ;LeftEdge
       dc.w 12         ;TopEdge
       dc.l TextAttr ;Font Attribute
       dc.l POT        ;ptr to String
       dc.l 0          ;NextText, if any for this gadget

myPotPropInfo1:
  dc.w 2             ;PotFlags = FREEHORIZ
  dc.w 0             ;HorizonPot
  dc.w 0             ;VertPot
  dc.w 16384         ;HorizonBody
  dc.w 0             ;VertPotBody
  dc.w 0,0,0,0,0,0   ;Intuition's Variables
myPotPropInfo2:
  dc.w 5             ;PotFlags = FREEVERT|AUTOKNOB
  dc.w 0
  dc.w 0
  dc.w 0
  dc.w 0
  dc.w 0,0,0,0,0,0

myPotImage1:
  dc.w  0,0  ;LeftEdge, TopEdge
  dc.w 48,6  ;Width, Height
  dc.w  1    ;Depth
  dc.l  0    ;ImageData (must point to a CHIP mem block.)
  dc.b  1    ;PlanePick
  dc.b  0    ;PlaneOnOff
  dc.l  0    ;NextImage
myPotImage2:
  ds.b  20   ;Since an AUTOKNOB, we don't initialize
             ;nor do we need any image data.

myStringInfo:
   dc.l   StrBuffer   ;Buffer
   dc.l   UndoBuffer  ;optional UndoBuffer
   dc.w   0           ;BufferPos
   dc.w   40          ;MaxChars (inc NULL) (you determine this)
   dc.w   0           ;DispPos
   dc.w   0,0,0,0,0   ;for Intuition
   dc.l   0
   dc.l   0           ;LongInt (for string integer gadg)
   dc.l   0           ;AltKeyMap (optional)

StrBuffer  ds.b 40  ;40 byte buffer
UndoBuffer ds.b 40  ;for undo (optional)
StrPts:
   dc.w   0,0,287,0,287,12,0,12,0,1,286,1,286,11,1,11,1,1
myStrBorder:
   dc.w   -4,-3    ;initial XY offsets, gadget relative
   dc.b   1,0      ;ForePen = 1, BackPen = 0
   dc.b   0        ;DrawMode
   dc.b   9        ;Count
   dc.l   StrPts   ;Vector
   dc.l   0        ;NextBorder


PotImageData:  ;This data needs to be in CHIP mem
  dc.w   -1,0,-1,-1,0,-1,0,-1,0
  dc.w   0,-1,0,-1,0,-1,-1,0,-1

POT           dc.b 'POT',0
```

•AC•

Tshell, part II

Enhancing the command line environment

by Rich Falconburg

This month I will continue coverage of Tshell, a new Amiga program that will enhance the command line environment. Because of the depth of this program, I have chosen to spread this review over two issues rather than just washing over the highlights in a single article. We continue this month with the path command.

The Tshell **PATH** command initially gets a list of the paths currently established by AmigaDOS. Directories added to the search tree using the Tshell PATH command will only be seen by the shell. The new paths are not returned to the operating system.

```
) path path_one path_two path_three
```

In the current implementation, there is no provision to remove a path from the list or change the order of priority.

The **PRI** command is similar to the AmigaDOS CHANGETASKPRI command, but it also lets you use the task ID number displayed by the TASKS command (covered later) to identify the process to change. This is especially handy for those programs that are begun in a detached CLI at a high priority. The Tshell TASKS command can display the ID number used by the system; the PRI command may be used to change its priority.

The **PRINT** command is a powerful facility for sending data to the printer.

PRINT Command Line Options

b	Boldface
c	Condensed
d	Double strike
e	Elite
E	Enlarged
i	Italics
l	Letter quality
p	Proportional font
h	Print the file name at the top of each page
s	Skip count at bottom of the page
q	Ignore escape sequences (see below)

A text file may be formatted with embedded escape characters to perform the functions shown above plus:

```
\s Subscript
\S Superscript
\u Underline
\n Restore to default mode (determined by command line parms)
```

The backslash character starts an Escape sequence.

If more than one print job is submitted, the others wait in a loop, checking the availability every ten seconds. If the printer is still busy after approximately 15 minutes, the job will exit and a message will be displayed. Unfortunately, a small procedure must be defined to execute print in the background as it does not do this by default. If no file name is given, the PRINT command may be used much like a typewriter.

The **PRINTF** command can be used to produce formatted output. The syntax of this command is similar to its C language equivalent. At present, only the %s specifier is operational.

The **PROCS** command lists currently defined alias procedures. This is handy for procedures you may define on-the-fly and would like to save to a file to be used later. By redirecting the output to a file, you can redefine the procedure by executing that file. Procedures, and script programming in general, are a complex part of Tshell. If there is any reader interest, let me know and I will cover the topic in greater detail in the future.

The **PS** command is similar to the AmigaDOS STATUS command. It produces nearly the same output. If the -l switch is supplied, the PS command will also display the task ID associated with each process.

The **PWD** command is used to display the current default directory. This is necessary because, as we learned last issue, the **CD** command entered alone will return us to the starting directory, or the HOME directory if defined. It is easiest to remember this command as "print working directory".

The **READ** command captures input from the keyboard and stores that data in a variable. The newline character (Carriage Return) is seen as the terminator.

The **RECEIVE** command adds a new dimension to the already robust Tshell environment. This command causes the current shell to wait for information on its public message port. This port may be used for interprocess communication and to pass information between CLI processes with the aid of the companion command, **SEND**. The SEND command uses the name of the public message port of another process to communicate. The IPC messages are implemented as ARexx-compatible RXCOMM messages with the RXFF_TOKEN flag set. String and function message types may be supported in a future release.

It is often convenient to have some means for passing multiple arguments to a command and to be able to selectively choose from a list of supplied values. The Tshell **SHIFT** command gives us this flexibility. Each supplied value is indicated by a numerical variable prefaced with a dollar sign. For example:

```
$0 $1 $2 $3 $4 $5...
```

POSITION ZERO is the name of the command issued. The positions that follow will hold the parameters used with the command. To select from this list you need only supply the SHIFT command with a positive or negative increment value such as:

```
) shift 3
```

This command also returns a value that may be used to determine the current position in the parameter list.

The **SLEEP** command is a simple way of pausing execution for a given number of seconds.

The **STACK** command is functionally similar to the AmigaDOS STACK command. If a value is given, the stack size will be altered to this value. Otherwise, the current stack size is displayed.

The following commands are similar to their C language counterparts. They are used to manipulate character strings in a variety of ways.

```
STRCAT      Joins strings together as one
STRCSPN     Returns the position of a character within a string
STRLEN      Returns the length of the specified string
STRSUB      Returns a substring within a string
```

By combining these commands in various ways, you can extract and manipulate information from variables. To exercise these (and at the same time provide me with some useful routines), I wrote some procedures to give me a UNIX-type of login facility. The following scripts are neither elegant nor very secure, but they do demonstrate some of the features available in Tshell. This provides me with several captive environments that set up aliases and defaults needed for several programs. Since I have hard drives on my system, I'm not too fond of the multiple start-sequence route. In a production environment, booting the system every so often is an unacceptable option. The procedures shown let me set up separate processes with specific characteristics.

The following file handles the logins:

```
cd SYS:
echo -n "\n\e[31mlogin: "
ulogin = 'read'
if (ulogin == "") ; login
echo -n "password: \e[30m"
upasswd = 'read'
uid = 'grep -f $ulogin s:passwd'
if (uid == "") ; echo "\e[31minvalid login" ; login
un = 'strsub uid (strcspn uid "!")'
pw = 'strsub uid (strcspn uid "!") (strcspn -n 2 uid "!")'
gp = 'strsub uid (strcspn -n 2 uid "!") (strcspn -n 3 uid "!")'
ud = 'strsub uid (strcspn -n 3 uid "!") (strcspn -n 4 uid "!")'
ds = 'strsub uid (strcspn -n 4 uid "!") (strcspn -n 5 uid "!")'
upl = 'strsub -t ud (strcspn ud "!")'
up = 'strsub upl (strcspn upl "!")'
if ((!$upasswd$! == pw) && (ulogin == un))
    {
    echo "\e[31m          Logged on to Amiga on" `date`
    echo "\n\n\n"
    clear -v "upasswd" "pw" "ulogin" "upl" "ud"
    receive -n un   # change the name of the port to the user
name
    cd up
    if test -e .profile ; .profile
    }
else
    {
    echo "\a\n\e[31minvalid login"
    login
    }
exit
```

The following file handles logging out:

```
echo "logged out on" `date`
if ($1 == "-c") ; exit ; echo `exit`
else ; s:login
```

The following file contains all the user information checked on each login. Its format is similar to that of a standard UNIX password file.

```
root!amiga!0!//!/dh0/c/tsh!
sys!pro!1!/dh0/system!/dh0/c/tsh!
wp!writer!100!/dh1/wordperfect/!/dh0/c/tsh!
sbp!gandalf!101!/dh2/superbase/!/dh0/c/tsh!
cxfer!xfer!200!/dh2/tp/vax_xfers/!/dh0/c/tsh!
cad!igr!201!/dh9/introcad/!/dh0/c/tsh!
art!painter!300!/dh9/paint/!/dh0/c/tsh!
dtp!writer!110!/dh1/PageStream/!/dh0/c/tsh!
mp!spread!400!/dh9/MaxiPlan/!/dh0/c/tsh!
plog!igr!110!/dh2/superbase/intergraph/!dh0/c/tsh!
```

The format used is:

```
login_name!password!group_number!home_directory!shell!
```

I have included a group classification for future use. The information in this file is compared to that entered. If a valid login and password combination is found, the login script will look for a .profile file in the directory defined in the home_directory field. If it exists, it is executed. Here is a sample .profile file for the wp login:

```
# This is a USER login file to set up user specific alias and
such.
#
HOME = DH1:WORDPERFECT
echo "\n\e[33m     Welcome to Amiga WordPerfect\n\n"

setfont Amiga 8
assign WP: $HOME
assign PRINT: WP
assign docs: WP:DOCUMENTS
assign books: Docs:Books
tcl := cd docs:general/articles

wp &
```

This file equates the HOME variable to the user directory, changes the font (WordPerfect gets ugly with anything other than an 8-bit font), and starts the program as a background process, indicated by the ampersand. This is much the same as saying RUN WP and is the recommended procedure for starting background processes. Why all this? First, I have a dedicated command interpreter window that places me in the directory of the program I am using. Second, with the HOME variable defined, I can move all over the disk and return to my login directory by simply entering "cd". The script then automatically sets up aliases, paths, and logical names needed for the program. Using this method, I can save on boot-up time by not having to assign every single thing each time I boot. Lastly, it's a simple way to establish a specific environment for different users—especially if they are not familiar with the computer.

The **TAIL** command is extremely useful when you are reviewing large text files and need to see something towards the end of the file. Normally, you would have to read through the entire file until you reach the part in which you are interested. Entering "tail" with an optional numeric value will cause the entered value of lines from the end of the file to be displayed. The default is 10. To see more (or less) enter a value such as:

```
) tail -50 test.c
```

This will print the last fifty lines of the file test.c.

The **TASKS** command lists the processes currently in the system. It is similar in some ways to the PS and AmigaDOS STATUS command except that it will list all tasks that are running. The other two commands will only display tasks associated with a CLI process. The AmigaDOS Exec is the controller of the multitasking operating system. All processes are Exec tasks but not all tasks are CLI processes. Confusing, huh? The TASKS command gives us a more realistic view of what is currently running in the system. It's curious that the author chose to provide this as a separate command rather than including a "-e" switch to the PS command that would do the same thing. This would keep the user interface closer to the UNIX shell environment that it emulates. This switch plus a "-f" and/or "- l" switch would make the PS command much more functional.

The **TEST** command is nearly as complete as its UNIX counterpart but has several important differences. This command may be used to determine several things about files, directories, and variables.

TEST Command Supported Switches

-s	Size of the file
-b	Number of disk blocks in the file
-t	Modification time and date of the file
-n	Returns TRUE if the argument is a number
-v	Returns TRUE if he argument is defined as a shell variable
-p	Returns TRUE if the argument is defined as an alias procedure
-e	Returns TRUE if the file or directory exists
-f	Returns TRUE if it is a file
-d	Returns TRUE if it is a directory
-x	Returns TRUE if the file is an executable (binary load file)
-R	TRUE if it exists and has Read permission
-W	TRUE if it exists and has Write permission
-E	TRUE if it exists and has Execute permission
-A	TRUE if the Archive bit is set
-P	TRUE if the Pure bit is set
-S	TRUE if the Script bit is set
-H	TRUE if the Hidden bit is set

As you can see, this command has significant utility. UNIX purists will argue that an existing shell script will not work as expected if this command is used. Agreed. This is true of many of the Tshell commands. I would prefer that a greater degree of compatibility with the UNIX shells be maintained. The properties that are necessarily unique to the Amiga could use either additional switches or new commands.

If you intend to use existing UNIX shell scripts with Tshell (and that's likely to be the case), you will spend a significant amount of time in conversion. This is unfortunate given the power available. If you are just looking for a more powerful script execution environment for your Amiga, I think you will be very happy with Tshell.

The **TIME** command may be used as a stopwatch. It will display the number of seconds that have elapsed since the last time it was reset with the "-r" switch.

The **TSH** command is used to invoke a new shell. The full command line specification is as follows:

```
tsh [-w] [-i[i] script] [-r ramdiskdir] [-s] [-S]
    [-Lpixel] [-Tpixel] [-Wpixel] [-Hpixel]
```

The brackets indicate optional parameters.

TSH Options

-w	Change the existing CLI window to a Tshell window. Take over this window.
-i	Start the shell and read "script", then pass control to the user
-ii	Same as -i, but exit at the end of the script
-r	Use a custom RAM disk command directory
-s	Get input from the standard input instead of interactively
-S	Append each line of the command history to the file tsh.out. (Useful with -ii above)
-L	Left pixel position of the shell window's top left corner
-T	Top pixel position of the shell window's top left corner
-W	Width of the shell window in pixels
-H	Height of the shell window in pixels

The "-i" and "-ii" may be used to define a specific startup file to be executed. The "-ii" method is useful mainly for background shells. Tshell will also look for three other files when invoked. If they exist, the shell will execute each one.

```
s:init.tsh [Only read when the first shell is started]
s:def.tsh [Always read]
s:ninit.tsh [Read as a default file for the "-i" switch]
```

The "-s" switch will cause the shell to get its input from somewhere other than the command line. This is useful for data coming from the SER: port or other device. Most of the other switches are self-explanatory.

The **VARS** command displays all currently defined shell variables. Since the variables are listed in the same manner used to define them, you can redirect the VARS command to a file, thus presenting a number of interesting possibilities.

Index of Amazing Advertisers

Discover something interesting? Need more information?

Please use the Reader Service Card in every issue of AC to contact those advertisers who have sparked your interest. Advertisers want to hear from you. This is the best way they have of determining the Amiga community's interests and needs. Take a moment and contact the companies with products you want to know more about. And, if you wish to contact an Amazing Advertiser directly, please tell them you saw their advertisement in:

Amazing COMPUTING™ AMIGA
Your Original AMIGA® Monthly Resource

Advertiser	Page	Reader Service Number
ACDA	92	104
AmiEXPO	65	115
Anivision	45	150
AROCK Computer Software	44	133
Arrakis	61	108
ASDG	101	112
B & B Computers	97	110
Central Coast Software	5	145
Computability	63	117
Consultron	90	156
Day's	103	171
E-Z Soft	96	193
Empire Graphics	80	151
Expansion Technologies	21	120
Express-Way Software	95	122
Great Valley Products	9	158
Joe's First Company	75	180
M.J. Systems	12	176
M.J. Systems	79	169
M2S	26	181
Micro Momentum	77	125
MicroBotics	13	109
Micromiga	70	182
Moonlight Development	6	190
Musicomp	40	159
NewTek	CIV	102
Omnitek Computers International, Inc.	54	136
One Byte	87	135
Pioneer Productions	40	183
Poor Person Software	49	127
Pre'spect Technics, Inc.	46	165
Safe Harbor	73	134
Sedona Software	30	119
Software Advantage Consulting Corp.	105	131
Software Integration Solutions	40	184
SunRize Industries	84	191
Supra	7	106
Tangent 270	2	153
Tensor Productions	55	141
The Bit Bucket Computer Store	107	139
The Memory Location	91	107

The **WC** command is used for counting characters, words, or lines in a file. Other utilities include IC and FC, an integer and floating-point calculator respectively, and DIMMER, a screen-blanking program. Both calculators have extensive command line operators and are designed for interactive computation.

I mentioned system variables last time, and lauded the configurability they allow the Tshell user. The following is a list of those variables and what they do:

```
AUTOINDENT   Indent procedures levels
COLORS       Allows escape sequences to change colors
DEFCD        Enables the "implied" cd command
HELP         Controls the HELP key operation
PAUSE        Controls the "more" facility
FILEOK       Ignores errors in scripts
PROCOK       Ignores errors in procedures
LINEOK       Continues execution on multi-command lines if a
             command fails
MAXPAND      Allows alias procedures in alias procedures
PROMPT       Command prompt
PROMPT2      Secondary prompt
HOME         Home directory
HUP          Cursor up control
HDOWN        Cursor down control
HNEXT        Next line control
HNUM         History buffer size
OBASE        Numeric base used
TS           Tab spacing value
TLBIN        Built-in command control flags
TLSYS        Code segment control flags


Read Only variables

_scr     Workbench screen pointer
_win     Tshell's window pointer
_rp      RastPort pointer
_con     ConUnit pointer
_maxx    Window width
_maxy    Window height
_maxc    Window width in characters
_maxr    Window height in characters
_curc    Cursor column position
_curr    Cursor row position
_random  A random positive 32-bit integer

System Environment Variables

tsh_initted  If not set, the file "s:init.tsh" will be
             executed by the first shell
BlockPen     Menu bar color
DetailPen    Menu bar text color
```

All these commands and their various capabilities combine with an environment that supports redirection and pipes, variable assignment, mathematical and boolean operators, flexible wild carding, multi-line alias and procedure construction with full decision control, background processing, function key assignments, system environment variables, startup initialization files, and an in-line editor. Surely all this is enough.

But wait! The Amiga is a graphics powerhouse. Shouldn't that graphics capability be accessible to the average CLI user? The people at Metran obviously think so. They have included two powerful commands making this possible. The **INTUITION** and **GRAPHICS** commands interface to the Amiga's ROM Kernel INTUITION and GRAPHICS libraries.

The INTUITION command functions include:

```
ActivateWindow _win
DisplayBeep _scr
MoveScreen _scr x y
MoveWindow _win x y
RefreshWindowFrame _win
ScreenToBack _scr
ScreenToFront _scr
SizeWindow _win x y
WBenchToBack
WBenchToFront
WindowLimits _win xmin ymin xmax ymax
WindowToBack _win
WindowToFront _win
```

The "_win" is a read-only variable that is a pointer to the shell's Intuition Window; "_scr" is a pointer to the Workbench Screen. The other arguments must be numeric.

The GRAPHICS command functions include:

```
ClearEOL _rp
ClearScreen _rp
Draw _rp x1 y1
DrawEllipse _rp cx cy rx ry
Move _rp x y
RectFill _rp x1 y1 x2 y2
SetAPen _rp fcolor
SetBPen _rp bcolor
SetRast _rp clrcolor
```

The "_rp" is a read-only variable that is a pointer to the Tshell Window's RastPort. The following script is one of the demo files included on the distribution disk and is a good example of the real power Tshell can provide.

```
# Check the arguments for validity
```

```
if ($# < 1) || (! (test -n $1))
{
    echo "`hilbert' draws an area-filling Hilbert fractal
curve"
    echo "usage: `hilbert level'"
    echo "The argument must be numeric; try `hilbert 3'"
    echo -n "Note: you must have the external `graphics'
command "
    echo "somewhere in your search path."
    exit -1
}

if $1 < 1; echo "argument must be > 0"; exit -2

# initialize some variables

# scale the drawing to fit inside the current window
blocks = ((1 << $1) - 1)  # the size of the figure, in line
segments
# width and height scaling numbers
sx = ((_maxx - 32) / blocks)
sy = ((_maxy - 34) / blocks)

if (sx < 1) || (sy < 1)
{
    # degenerate figure; height or width of 0 pixels
    clear -v "sx" "sy" "blocks"
    echo "try a smaller argument or a larger window"
    exit -1
}

Px = 8; Py = 14   # the current position (upper left corner of
Tshell
window)
Vx = 1; Vy = 0    # "velocity": the distance to move at this
step
color = 2     # the graphics color to use when drawing the line

# make ROM kernal graphics interface AmigaDOS resident

load -a graphics
graphics Move _rp Px Py

# get cursor to bottom of window, then print a message

clear -w          # clear the window; puts cursor on top line
while _curr < _maxr; echo # prints newlines until at the last
line
# print a label only if it will fit without wrapping to the
next line
label = "Hilbert figure "
if _maxc > (3 + (strlen label)); echo -n label
clear -v "label" # undefine label; we don't need it any more

# Now define some procedures:

# procedure "go"
# cycle the graphics color and draw to the new position

go :=
{
    if (++ color) > 3; color = 1
    graphics SetAPen _rp color
    graphics Draw _rp (Px += (Vx * sx)) (Py += (Vy * sy))
}

# procedure "turn"; determine new velocity values for next line
# $1 is distance; direction is determined based on current
direction
# (the direction that is affected is the one that is now zero)

turn :=
{
    if Vx == 0
    {
        if Vy > 0; Vx = $1
        else Vx = (- $1)
        Vy = 0
    }
    else
    {
        if Vx < 0; Vy = $1
        else Vy = (- $1)
        Vx = 0
    }
}

# procedure "H"
# $1 = level, $2 = direction
# draw the next level of the figure; recurses

H :=
{
    if $1 == 0; return

    turn (- $2)
    H ($1 - 1) (- $2)
    go
    turn $2
    H ($1 - 1) $2
    go
    H ($1 - 1) $2
    turn $2
    go
    H ($1 - 1) (- $2)
    turn (- $2)
}

# draw the whole thing
H $1 1

# cleanup; clear definitions of procedures
clear -p go turn H
# be nice on memory usage; unload the graphics command
load -au graphics
```

Tshell includes a powerful editor that has the advantage of being part of the command line environment. It's difficult to visualize how it operates without actually using it. Pressing Control E turns the editor on and off. When on, the history buffer can be manipulated with other control sequences.

History Buffer Control Sequences

CTRL T	Tag this line
CTRL G	Go to the tagged linCTRL R Read a file into the buffer
CTRL W	Write the buffer from the tag to the cursor to a file
CTRL K	Delete from cursor to end of the line
CTRL X	Delete the line and leave a space
CTRL D	Delete the line and close the space
CTRL P	Insert text deleted with CTRL X OR D below cursor
CTRL O	Open a line above the cursor
CTRL S	Execute lines from cursor to tag
CTRL Z	Toggle display overwrite flag
CTRL C	Abort

There is still much more that could be said about this program, but I have to stop somewhere. Tshell author Jay Ts is continually adding features and functions that will make future releases even more powerful. One item on the top of my list is AUX: support. Jay assures me this is in the works. The design of the editor and the character-oriented operation makes this a very attractive option. More compatibility with Unix shell syntax would greatly improve the program's attractiveness. If there is enough interest, I will cover more of this program in greater detail in future issues. Next time I will begin coverage of the new Workbench 1.3 features and additions. •AC•

Metran Technology
1907 Briar Ave., Utica NY
(315) 732-4558
Tshell: *$50.00*
Inquiry #200

Function Evaluator *in C*

by Randy Finch

In the process of converting a 3D function plotting program from BASIC to C, I realized I would have to write a function evaluator routine similar to what any spreadsheet would have. To plot different functions in the BASIC version of the program, it is possible to stop the program, edit the function, and then rerun the program. In a compiled language such as C, however, this is not possible. I needed a C routine that would accept a mathematical function as a string input and then evaluate the function at different values. After perusing a Modula-2 program demonstrating this technique in the book *Software Engineering With Modula-2 and Ada* by Weiner and Sincovec (John Wiley & Sons, Inc., 1984), I began writing a similar but more extensive routine in C. (Note: In order to avoid conflicts in terminology when referring to C functions and mathematical functions, I will hereafter refer to mathematical functions as equations.)

The routine that resulted from my efforts consists of two externally accessible functions and many static support functions. All these functions reside in a file entitled FUNCEVAL.C, which is shown in Listing One. The two externally accessible functions are called Convert and Evaluate. Convert transforms a string containing a two-variable (X and Y) mathematical equation in standard notation into another string that is in a notation that can be evaluated more quickly and efficiently (the function Convert is near the end of Listing One). Evaluate accepts two double-precision floating-point numbers and returns the value of the equation upon substituting these two values for X and Y.

The parameter FunctionString is a pointer to a null-terminated string of unsigned characters. This string should contain an equation in two variables—X and Y—and may contain any of several transcendental and other function calls such as SIN, LN, SQRT, etc. (a complete list of acceptable function calls and mathematical operators is shown in Table One). FunctionString should be in standard mathematical notation, as the following equation illustrates:

```
-12.5 + SIN(X^2) / COS(LN(Y) + 2.2e-1)
```

FunctionString can contain any number of spaces and the alphabetic characters used for function and variable names can be in either upper or lower case. The first two support functions called by Convert—RemoveSpaces and strupr—remove all the spaces in the string and convert all lower case letters to upper case.

CheckSyntax, the next function called by Convert, examines the syntax of FunctionString. CheckSyntax can detect 12 different syntax errors, including illegal characters, misplaced operators, and missing delimiters. These errors are symbolically defined in the header file SYNTXERR.H, which is shown in Listing Two. If an error is detected, the externally accessible global variable SyntaxErr is equated with an appropriate error value and a pointer to the offending character in FunctionString is then returned. If no error is found, SyntaxErr is set to FALSE and a zero is returned.

After FunctionString passes the syntax check, it is copied to another string, fstr. This allows the original form of the equation (with spaces removed and all alphabetic characters in upper case) to be preserved in FunctionString while modifications are made to the copy in fstr.

The first modification to fstr is performed by the support function ConvertConstants. This function initially scans fstr for unary pluses and minuses and places a zero in front of each sign. This action will make further processing of the string easier. Next, ConvertConstants scans fstr once again, this time replacing all numeric constants with a one-byte symbol (within a range of 128 to 255) and placing the actual value of the constant in an array for future reference. The maximum number of constants allowed in the string is 128. If fstr contains more than 128 constants, SyntaxErr is set to TOOMANYCONSTS and FALSE is returned. Otherwise, TRUE is returned.

The next function called by Convert, ConvertFunctions, scans fstr and replaces all transcendental and other function names with a one-byte symbol within a range of 1 to 13. These symbols are defined at the beginning of FUNCEVAL.C. No error checking is necessary in this function because illegal functions will have already been detected in CheckSyntax. Therefore, ConvertFunctions is a void function.

The last support function called by Convert, InfixToPostfix, converts fstr—which is still in standard mathematical notation and has its constants and function names substituted with one-byte symbols—into a postfix notation. This is the notation used by the computer language Forth and by Hewlett-Packard calculators. InfixToPostfix is a relatively complex function that reads the string fstr character by character and, based upon the precedence of the operators, either places the character in the static global string NewExpr or places it on a stack for later processing. If the stack underflows or overflows during processing, SyntaxErr will be equated with STACKUNDERFLOW or STACKOVERFLOW and FALSE will be returned. When processing is complete, NewExpr will contain the mathematical equation in postfix notation and TRUE will be returned.

The Convert and ConvertConstants functions have several lines of code which will only be compiled if the symbol DEBUG is defined. These sections of code will print the equation at

various points during its processing. Let's look at an example step by step.

Suppose the function is originally entered by the user as follows:

```
2.1E-1 + x*y - sin(-.8*X) / ln(+Y + 100)
```

The following steps will convert this function into symbolic form with postfix notation (the support function that accomplishes each step appears in parentheses):

STEP 1. Remove all spaces. (RemoveSpaces)

```
2.1E-1+x*y-sin(-.8*X)/ln(+Y+100)
```

STEP 2. Convert lower case letters to upper case. (strupr)

```
2.1E-1+X*Y-SIN(-.8*X)/LN(+Y+100)
```

STEP 3. Check syntax. If no errors are detected, processing continues. No changes are made to the string. (CheckSyntax)

STEP 4a. Put zeroes before unary pluses and minuses. (ConvertConstants)

```
2.1E-1+X*Y-SIN(0-.8*X)/LN(0+Y+100)
```

STEP 4b. Replace constants with one-byte symbols within a range of 128 to 255. Since these are non-printable characters, each character in the string will be printed below as a decimal value. The actual character or constant it represents appears beneath the decimal value. (ConvertConstants)

```
128  43  88  42  89  45  83  73  78  40  129  45  130  42
88…
21.E-1 + X   *   Y   -   S   I   N   (   0   -   .8   *
X…

…41  47  76  78  40  131  43  89  43  132  41
 … ) /   L   N   (   0   +   Y   +   100   )
```

STEP 5. Replace function names with one-byte symbols within a range of 1 to 13. (ConvertFunctions)

SIN, represented in STEP 4 by the sequence [83 73 78], becomes simply [1].

LN, represented in STEP 4 by the sequence [76 78], simply becomes [12].

STEP 6. Convert the string to postfix notation. (InfixToPostfix)

```
 128    88    89    42    43    129    130    88    42
2.1E-1   X     Y     *     +     0      .8     X     *

45     1    131    89    43    132    43    12    47    45
 -    SIN    0     Y     +    100     +    LN     /     -
```

The final form of the equation in this step resides in the string NewExpr. A pointer to this string is returned by Convert.

The function Evaluate (at the end of Listing One) can be used to evaluate the equation at different values of X and Y. Because the equation is now in postfix notation, Evaluate is a rather simple routine. It simply reads the characters in NewExpr sequentially. If a character represents X, Y, or a constant, its actual numerical value is pushed onto a stack. If a character represents a function, Evaluate will pass the number on the top of the stack, along with the character symbol representing the function, to the Calculate function. The result of applying the number to this function will be returned. For example, if the number equals 100 and the function is LOG, a value of 2 will be returned. This return value is then pushed onto the top of the stack.

If a character represents an operator (^, +, -, *, or /), the top two numbers on the stack and the operator symbol are passed to Calculate. The result of performing the operation on the two numbers is returned. For example, if the number on top of the stack is 24, the next number on the stack is 12, and the operator is /, Calculate will return 12/24, or 0.5. Notice that the operation is always performed in the following order:

(Second number on stack) Operator (Top number on stack)

The above procedure continues until no more characters remain to be read from NewExpr. At this point the value of the equation with the appropriate values of X and Y substituted will be on top of the stack. This value is returned by Evaluate. Please note that Evaluate does not check for illegal mathematical operations such as the square root of a negative number or a multiplication that causes an overflow. The methods for handling these types of errors vary with different compilers. It is left up to the reader to add this form of error handling (if it is needed).

Table One: Function Calls and Mathematical Operators Acceptable to FUNCEVAL.C

Functions

- SIN (sine)
- COS (cosine)
- TAN (tangent)
- ASIN (arcsine)
- ACOS (arccosine)
- ATAN (arctangent)
- SINH (hyperbolic sine)
- COSH (hyperbolic cosine)
- TANH (hyperbolic tangent)
- EXP (exponential)
- SQRT (square root)
- LN (natural logarithm)
- LOG (logarithm base 10)

Operators

- ^ (raise to power)
- * (multiplication)
- / (division)
- + (addition)
- - (subtraction)
- ( ) (used to group operations)

Note: The standard hierarchy of operators prevails.

A program entitled TESTFEVL.C is shown in Listing Three. This program tests the Convert and Evaluate functions. The user is allowed to input a mathematical equation (up to 255 characters long) along with values of X and Y. The program will print the result of evaluating the equation. Unfortunately, the scanf function (from the C library) does not allow spaces to be entered in a string, as they are used for delimiters; therefore, you will not be able to see the RemoveSpaces function in action. (Note: If spaces are entered as part of the equation, the program will behave erratically as characters after the spaces are used as input for later scanf calls.)

If compiling FUNCEVAL.C with DEBUG defined, each stage of the equation conversion will be printed to the standard output. You can play to your heart's content. Make sure some equations that are syntactically incorrect are typed in to see how the error detection works. The equation will be printed, an arrow will appear below the offending character, and an appropriate error message will be printed.

FUNCEVAL.C and TESTFEVL.C are written generically and have been successfully compiled, linked, and executed on an Amiga 1000 using Lattice C v4.01 and v5.02, on an IBM AT using Microsoft C v4.0, and on a Cray XcM/P supercomputer using the Cray C compiler (with one modification). If you are using an older C compiler, some of the standard library functions used in this program may not be available. If you have any questions regarding the information presented in this article, you may write to Randy Finch, c/o Amazing Computing, P.O. Box 869, Fall River, MA 02722.

Listing One: FUNCEVAL.C

```
/*********************************************************
*                                                        *
*  Convert() converts a string containing a math         *
*  function in two variables, X and Y, to a postfix      *
*  notation string with the numeric constants and        *
*  functions converted to one byte symbols.  If any      *
*  syntax errors occur they will be posted in the global *
*  external variable SyntaxErr.  Definitions are in      *
*  syntxerr.h.                                           *
*                                                        *
*  Evaluate() will substitute the X and Y values passed  *
*  to it and return the value of the function.           *
*                                                        *
*  This program may be freely used for non-profit        *
*  purposes as long as the copyright notice remains      *
*  in the code.                                          *
*                                                        *
*********************************************************
*                                                        *
*  funceval.c v3.0 - Copyright 1988  Randy C. Finch      *
*                                                        *
*********************************************************/

/*--------  INCLUDES  --------*/

#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <float.h>
#include <stdlib.h>
#include "syntxerr.h"

/*--------  DEFINES  --------*/

#define NUMSYM 128      /* Number of constants allowed in
function */
#define SYMBASE 128     /* Base value for constants symbols */
#define STACKSIZE 256   /* Stack size */
#define SIN  1          /* Symbol for sine */
#define COS  2          /* Symbol for cosine */
#define TAN  3          /* Symbol for tangent */
#define ASIN 4          /* Symbol for arcsine */
#define ACOS 5          /* Symbol for arccosine */
#define ATAN 6          /* Symbol for arctangent */
#define SINH 7          /* Symbol for hyperbolic sine
#define COSH 8          /* Symbol for hyperbolic cosine */
#define TANH 9          /* Symbol for hyperbolic tangent */
#define EXP  10         /* Symbol for exponential */
#define SQRT 11         /* Symbol for square root */
#define LN   12         /* Symbol for natural logarithm */
#define LOG  13         /* Symbol for logarithn base 10 */
#define NULL 0          /* Symbol for null */
#define TRUE 1          /* Symbol for true condition */
#define FALSE 0         /* Symbol for false condition */


/*--------  EXTERNAL GLOBALS  --------*/

unsigned char SyntaxErr;


/*--------  GLOBALS  --------*/

struct CharStack {
  unsigned char c[STACKSIZE];
  long          top;
};

struct NumStack {
  double n[STACKSIZE];
  long   top;
};
```

```
static struct CharStack cstack;

static struct NumStack nstack;

static double Constants[NUMSYM];

static unsigned char CurConstant;

static unsigned char NewExpr[256];


/*--------  FUNCTIONS  --------*/

static char CharInStr(s,c)
  unsigned char *s;
  unsigned char c;
{
  while (*s != NULL) {
     if (*s == c) return TRUE;
     ++s;
  }
  return FALSE;
} /* CharInStr */

static void Deposit(num)
  double num;
{
  Constants[CurConstant - SYMBASE] = num;
} /* Deposit */


static void Substitute(symb, ptr, len)
  unsigned char symb;
  unsigned char *ptr;
  unsigned long len;
{
  *ptr = symb;

  if (len > 1) {
     do {
        ++ptr;
        *ptr = *(ptr + len - 1);
     } while (*ptr != NULL);
  }
} /* Substitute */


static void RemoveSpaces(str)
  unsigned char *str;
{
  unsigned char *ptr;

  while (*str != NULL) {
     if (*str == ' ') {
        ptr = str;
        do {
           *ptr = *(ptr + 1);
           ++ptr;
        } while ( *(ptr - 1) != NULL);

        --str;
     }

     ++str;
  }
} /* RemoveSpaces */


static void AddZero(ptr)
  char *ptr;
{
  unsigned long len;
  char *i;
  len = strlen(ptr);

  for (i=ptr+len+1; i>ptr; --i)
     *i = *(i - 1);

  *ptr = '0';
} /* AddZero */
```

```
static unsigned char CPop()
{
  if (cstack.top == 0) return 0;
  else {
     --cstack.top;
     return cstack.c[cstack.top + 1];
  }
} /* CPop */


static char CPush(c)
  unsigned char c;
{
  if (cstack.top == STACKSIZE) return FALSE;
  else {
     ++cstack.top;
     cstack.c[cstack.top] = c;
     return TRUE;
  }
} /* CPush */


static unsigned char CTopOfStack()
{
  return cstack.c[cstack.top];
} /* CTopOfStack */


static double NPop()
{
  --nstack.top;
  return nstack.n[nstack.top + 1];
} /* NPop */


static void NPush(n)
  double n;
{
  ++nstack.top;
  nstack.n[nstack.top] = n;
} /* NPush */

static char IsFunction(c)
  unsigned char c;
{
  if ( (c >= SIN) && (c <= LOG) )
     return TRUE;
  else
     return FALSE;
} /* IsFunction */


static char IsSymbol(c)
  unsigned char c;
{
  if ((c >= SYMBASE) && (c < SYMBASE+NUMSYM))
     return TRUE;
  else
     return FALSE;
} /* IsSymbol */


static char Precedence(c1,c2)
  unsigned char c1,c2;
{
  if ( (CharInStr("+-*/",c1)) && (c2 == '^') )
     return FALSE;

  else if ( (CharInStr("+-",c1)) && (CharInStr("*/",c2)) )
     return FALSE;

  else if ( ((c1 == '(') && (c2 != ')')) || (c2 == '(') )
     return FALSE;

  else if ( (CharInStr("+-*/^",c1)) && (IsFunction(c2)) )
     return FALSE;

  else
     return TRUE;
} /* Precedence */
```

```
static unsigned char *CheckSyntax(str)
  unsigned char *str;
{
  int numLP = 0,
      numRP = 0;

  if ( (CharInStr("/*^E)",*str)) && (strncmp(str,"EXP",3) != 0)
) {
     if (CharInStr("/*^",*str)) {
        SyntaxErr = MISPLACEDOP;
        return str;
     }
     else if (*str == 'E') {
        SyntaxErr = ILLEGALEXP;
        return str;
     }
     else {
        SyntaxErr = MISSINGLP;
        return str;
     }
  } /* if */

  for (;;) {   /* forever */

     if (*str == '(') {
        ++numLP;
        ++str;
        if ( (CharInStr("*/^E",*str)) && (strncmp(str,"EXP",3)
!= 0) ) {
           if (*str == 'E') {
              SyntaxErr = ILLEGALEXP;
              return str;
           }
           else {
              SyntaxErr = MISPLACEDOP;
              return str;
           }
        } /* if */
        if ( (*str == ')') || (*str == NULL) ) {
           SyntaxErr = MISSINGPARM;
           return str;
        }
     } /* if */

     else if (*str == ')') {
        ++numRP;
        ++str;
        if (numRP > numLP) {
           SyntaxErr = MISSINGLP;
           return (str-1);
        }
        else if ( (!CharInStr(")+-*/^",*str)) && (*str != NULL)
) {
           SyntaxErr = MISSINGOP;
           return str;
        }
     } /* else if */

     else if ( (isdigit(*str)) || (*str == '.') ) {
        char ExitFlag = FALSE,
             OneDecimal = FALSE,
             OneE = FALSE;
        if (*str == '.') OneDecimal = TRUE;

        ++str;

        if ( (OneDecimal == TRUE) && (!isdigit(*str)) ) {
           SyntaxErr = LONEDECIMAL;
           return (str - 1);
        }

        while ( ( (isdigit(*str)) || (CharInStr(".E)-+",*str))
                || (*str == NULL) ) && !ExitFlag ) {

           if (*str == '.') {
              ++str;
              if (OneE) {
                 SyntaxErr = ILLEGALEXP;
                 return (str-1);
              }
              else if (OneDecimal) {
                 SyntaxErr = EXTRADECIMAL;
                 return (str-1);
              }
              else if (strncmp(str,"EXP",3) == 0) {
                 SyntaxErr = MISSINGOP;
                 return str;
              }
              else if ( (!CharInStr("+-*/^E)",*str)) &&
(!isdigit(*str)) ) {
                 SyntaxErr = ILLEGALCHAR;
                 return str;
              }
              else {
                 OneDecimal = TRUE;
              }
           } /* if */

           else if (*str == 'E') {
              ++str;
              if (OneE) {
                 SyntaxErr = EXTRAE;
                 return (str-1);
              }
              else if ( (!CharInStr("+-",*str)) &&
(!isdigit(*str)) ) {
                 SyntaxErr = ILLEGALEXP;
                 return str;
              }
              else {
                 OneE = TRUE;
              }
           } /* else if */

           else if (CharInStr("+-",*str)) {
              if ( *(str-1) == 'E' )
                 ++str;
              else if ( !OneE || (OneE && isdigit(*(str-1))) )
```

```
            ExitFlag = TRUE;
         else {
            SyntaxErr = MISPLACEDOP;
            return str;
         }
      } /* else if */

      else if ( (*str == ')') || (*str == NULL) ) {
         if (CharInStr("+-E",*(str-1))) {
            SyntaxErr = ILLEGALEXP;
            return str;
         }
         else {
            ExitFlag = TRUE;
         }
      } /* else if */

      else {
         ++str;
      } /* else */

   } /* while */

   if( !CharInStr("+-*/)", *str) && (*str != NULL) ) {
      SyntaxErr = MISSINGOP;
      return str;
   }

} /* else if */

else if (CharInStr("+-*/^",*str)) {
   ++str;
   if ( (CharInStr(")E+-*/^",*str)) || (*str == NULL) ) {
      if (strncmp(str,"EXP",3) != 0) {
         SyntaxErr = MISPLACEDOP;
         return (str-1);
      }
   }
} /* else if */

else if (CharInStr("XY",*str)) {
   ++str;
    if ( (!CharInStr(")+-*/^",*str)) && (*str != NULL)
) {
      SyntaxErr = MISSINGOPRP;
      return str;
   }
} /* else if */
else if (isupper(*str)) {
   if (strncmp(str,"LN",2) == 0) str += 2;
   else if (strncmp(str,"SINH",4) == 0) str += 4;
   else if (strncmp(str,"COSH",4) == 0) str += 4;
   else if (strncmp(str,"TANH",4) == 0) str += 4;
   else if (strncmp(str,"SIN",3) == 0) str += 3;
   else if (strncmp(str,"COS",3) == 0) str += 3;
   else if (strncmp(str,"TAN",3) == 0) str += 3;
   else if (strncmp(str,"EXP",3) == 0) str += 3;
   else if (strncmp(str,"LOG",3) == 0) str += 3;
   else if (strncmp(str,"SQRT",4) == 0) str += 4;
   else if (strncmp(str,"ASIN",4) == 0) str += 4;
   else if (strncmp(str,"ACOS",4) == 0) str += 4;
   else if (strncmp(str,"ATAN",4) == 0) str += 4;
   else {
      SyntaxErr = ILLEGALFUNC;
      return str;
   }

   if (*str != '(') {
      SyntaxErr = MISSINGLP;
      return str;
   }
} /* else if */

else if (*str == NULL) {
   if (numLP < numRP) {
      SyntaxErr = MISSINGLP;
      return str;
   }
   else if (numLP > numRP) {
      SyntaxErr = MISSINGRP;
      return str;
   }
   else {
      SyntaxErr = FALSE;
      return 0L;
   }
} /* else if */

else {
   SyntaxErr = ILLEGALCHAR;
   return str;
}

} /* for */

} /* CheckSyntax */


static char ConvertConstants(str)
  unsigned char *str;
{
  unsigned char *ptr;

  ptr = str;
  if ( CharInStr("+-",*ptr) ) {
     AddZero(str);
     ptr += 2;
  }

  while ( *ptr != NULL ) {
     if ( (CharInStr("+-",*ptr)) && (*(ptr-1) == '(') )
        AddZero(ptr);

     ++ptr;

  } /* while */

#if DEBUG
  printf("\nAddZero: %s\n", str);
#endif

  {  /* begin block */
  unsigned long j;
  unsigned char numstr[80];
  double number;

  ptr = str;

  CurConstant = SYMBASE;

  while ( *ptr != NULL) {
     if ( (*ptr == '.') || (isdigit(*ptr)) ) {
        unsigned long lennum = 1;

        while ( (CharInStr(".E-+",*(ptr+lennum))) ||
(isdigit(*(ptr+lennum))) ) {
           if( (CharInStr("-+",*(ptr+lennum))) &&
(*(ptr+lennum-1) != 'E') )
              break;
           ++lennum;
        }

        for (j=0; j<lennum; ++j)
           *(numstr+j) = *(ptr+j);

        *(numstr+j) = NULL;

        number = atof(numstr);
        Deposit(number);
        Substitute(CurConstant, ptr, lennum);

        ++CurConstant;
        if (CurConstant >= SYMBASE+NUMSYM) {
           SyntaxErr = TOOMANYCONST;
           return FALSE;
        }

     } /* if */

     ++ptr;
```

```
  } /* while */

  }  /* end block */

  return TRUE;

} /* ConvertConstants */


static void ConvertFunctions(str)
  unsigned char *str;
{
  while ( *str != NULL ) {
     if ( (isupper(*str)) && (!CharInStr("XY",*str)) ) {

        if (strncmp(str,"LN",2) == 0) Substitute(LN,str,2L);
        else if (strncmp(str,"SINH",4) == 0)
Substitute(SINH,str,4L);
        else if (strncmp(str,"COSH",4) == 0)
Substitute(COSH,str,4L);
        else if (strncmp(str,"TANH",4) == 0)
Substitute(TANH,str,4L);
        else if (strncmp(str,"SIN",3) == 0)
Substitute(SIN,str,3L);
        else if (strncmp(str,"COS",3) == 0)
Substitute(COS,str,3L);
        else if (strncmp(str,"TAN",3) == 0)
Substitute(TAN,str,3L);
        else if (strncmp(str,"EXP",3) == 0)
Substitute(EXP,str,3L);
        else if (strncmp(str,"LOG",3) == 0)
Substitute(LOG,str,3L);
        else if (strncmp(str,"SQRT",4) == 0)
Substitute(SQRT,str,4L);
        else if (strncmp(str,"ASIN",4) == 0)
Substitute(ASIN,str,4L);
        else if (strncmp(str,"ACOS",4) == 0)
Substitute(ACOS,str,4L);
        else if (strncmp(str,"ATAN",4) == 0)
Substitute(ATAN,str,4L);

     } /* if */

     ++str;

  } /* while */

} /* ConvertFunctions */


static char InfixToPostfix(str)
  unsigned char *str;
{
  unsigned long i1=0, i2=0;
  unsigned char NextChar, TopSymbol;

  cstack.top = 0;       /* Initialize stack */
  NewExpr[0] = NULL;    /* Initialize expression */

  while ( *(str+i1) != NULL ) {

     NextChar = *(str+i1);

     if ( (IsSymbol(NextChar)) || (NextChar == 'X') ||
(NextChar == 'Y') ) {
        NewExpr[i2] = NextChar;
        ++i2;
     }
     else {
        for (;;) {   /* Forever */

           if ( (cstack.top == 0) ||
(!Precedence(CTopOfStack(),NextChar)) )
              break;

           if ((TopSymbol = CPop()) == 0) {
              SyntaxErr = STACKUNDERFLOW;
              return FALSE;
           }

           if (cstack.top != 0) {
              if ( (IsFunction( CTopOfStack() )) && (NextChar
== ')') ) {
                 TopSymbol = CPop();
                 NewExpr[i2] = TopSymbol;
                 ++i2;
                 break;
              } /* if */
           } /* if */

           if ( (TopSymbol == '(') && (NextChar == ')') )
              break;

           if (TopSymbol != '(') {
              NewExpr[i2] = TopSymbol;
              ++i2;
           }

        } /* for */

        if (NextChar != ')') {
           if (CPush(NextChar) == FALSE) {
              SyntaxErr = STACKOVERFLOW;
              return FALSE;
           }
        }

     } /* if */
     ++i1;

  } /* while */

  while (cstack.top != 0) {
     TopSymbol = CPop();
     if (TopSymbol != '(') {
        NewExpr[i2] = TopSymbol;
        ++i2;
     }
```

```
  }

  NewExpr[i2] = NULL;

  return TRUE;

} /* InfixToPostfix */


static double Calculate(s,n2,n1)
  unsigned char s;
  double n1,n2;
{
  switch (s) {
     case '+':
        return (n1 + n2);
     case '-':
        return (n1 - n2);
     case '*':
        return (n1 * n2);
     case '/':
        return (n1 / n2);
     case '^':
        return ( exp(n2*log(n1)) );
     case SIN:
        return ( sin(n2) );
     case COS:
        return ( cos(n2) );
     case TAN:
        return ( tan(n2) );
     case EXP:
        return ( exp(n2) );
     case SQRT:
        return ( sqrt(n2) );
     case LN:
        return ( log(n2) );
     case LOG:
        return ( log10(n2) );
     case ASIN:
        return ( asin(n2) );
     case ACOS:
        return ( acos(n2) );
     case ATAN:
        return ( atan(n2) );
     case SINH:
        return ( sinh(n2) );
     case COSH:
        return ( cosh(n2) );
     case TANH:
        return ( tanh(n2) );

  } /* switch */

} /* Calculate */


unsigned char *Convert(FunctionString)
  unsigned char *FunctionString;
{
  unsigned char fstr[512];
  unsigned char *ptr;

  SyntaxErr = FALSE;

  RemoveSpaces(FunctionString);

#if DEBUG
  printf("\nRemoveSpaces: %s\n", FunctionString);
#endif

  strupr(FunctionString);

#if DEBUG
  printf("\nstrupr: %s\n", FunctionString);
#endif

  if ((ptr = CheckSyntax(FunctionString)) != 0) return ptr;

  strcpy(fstr,FunctionString);

#if DEBUG
  printf("\nCheckSyntax: %s\n", fstr);
#endif

  if (!ConvertConstants(fstr)) return fstr;

#if DEBUG
  printf("\nConvertConstants: ");

   for(ptr = fstr; *ptr != NULL; ++ptr)
     printf("%d ", *ptr);
  printf("\n");
#endif

  ConvertFunctions(fstr);

#if DEBUG
  printf("\nConvertFunctions: ");

  for(ptr = fstr; *ptr != NULL; ++ptr)
     printf("%d ", *ptr);

  printf("\n");
#endif

  if (!InfixToPostfix(fstr)) return fstr;

#if DEBUG
  {
  unsigned char *ptr;

  printf("\nInfixToPostfix: ");

  for(ptr = NewExpr; *ptr != NULL; ++ptr)
     printf("%d ", *ptr);

  printf("\n");
  }
#endif

  return NewExpr;

} /* Convert */


double Evaluate(x,y)
  double x, y;
{
  unsigned char symbol;
  long i = 0;

   nstack.top = 0;   /* Initialize stack */

   while (NewExpr[i] != NULL) {
      symbol = NewExpr[i];

      if (symbol == 'X')
         NPush(x);

      else if (symbol == 'Y')
         NPush(y);
      else if (IsSymbol(symbol)) {
         NPush( Constants[symbol-SYMBASE] );
      }

      else if (IsFunction(symbol)) {
         NPush( Calculate(symbol, NPop(), 0.0) );
      }

      else {
         NPush( Calculate(symbol, NPop(), NPop()) );
      }

      ++i;

   } /* while */

   return NPop();

} /* Evaluate */
```

Listing Two: SYNTXERR.H

```
/*********************************************************
*                                                       *
*      Definitions for syntax errors that may occur in  *
*      the function Convert in funceval.c.              *
*                                                       *
*********************************************************
*                                                       *
*      syntxerr.h - Copyright 1988  Randy C. Finch      *
*                                                       *
*********************************************************/

#define MISPLACEDOP      1     /* Misplaced operator */
#define ILLEGALCHAR      2     /* Illegal character */
#define ILLEGALEXP       3     /* Illegal exponent */
#define ILLEGALFUNC      4     /* Illegal function */
#define MISSINGOP        5     /* Missing operator */
#define MISSINGOPRP      6     /* Missing operator or right
parenthesis  */
#define MISSINGLP        7     /* Missing left parenthesis */
#define MISSINGRP        8     /* Missing right parenthesis
*/
#define MISSINGPARM      9     /* Missing parameter */
#define LONEDECIMAL     10     /* Lone decimal point */
#define EXTRADECIMAL    11     /* Extra decimal point */
#define EXTRAE          12     /* Extra E in exponent */
#define STACKUNDERFLOW  13     /* Stack underflow */
#define STACKOVERFLOW   14     /* Stack overflow */
#define TOOMANYCONST    15     /* Too many constants in
function  */
```

Listing Three: TESTFEVL.C

```
/*********************************************************
*                                                       *
*  This program tests the functions Convert and         *
*  Evaluate in funceval.c.  It will allow the user to   *
*  type in a function (no spaces please) and then       *
*  enter values for X and Y.  The value of the function *
*  with these values of X and Y substituted will be     *
*  printed.  If an error is found in the function, the  *
*  function will be printed to the standard output, an  *
*  up arrow will be printed beneath the offending       *
*  character and an appropriate error message will be   *
*  printed.                                             *
*                                                       *
*  This program may be freely used for non-profit       *
*  purposes as long as the copyright notice remains     *
*  in the code.                                         *
*                                                       *
*********************************************************
*                                                       *
*  testfevl.c - Copyright 1988  Randy C. Finch          *
*                                                       *
*********************************************************/

/*--------   INCLUDES   --------*/

#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <float.h>
#include "syntxerr.h"


/*--------   DEFINES   --------*/

#define NULL 0


/*--------   EXTERNAL VARIABLES   --------*/

extern double Evaluate();
extern unsigned char *Convert();
extern unsigned char SyntaxErr;


/*--------   FUNCTIONS   --------*/
void main()
{
  unsigned char function[256];
  unsigned char *expression;
   unsigned long spaces, i;
   char answer1[10], answer2[10];

  do {

     printf("\nEnter a function:\n\n");
     scanf("%s",function);

     expression = Convert(function);

     if (SyntaxErr) {
        if (SyntaxErr == STACKUNDERFLOW)
           printf("Stack underflow\n");
        else if (SyntaxErr == STACKOVERFLOW)
           printf("Stack overflow\n");
        else if (SyntaxErr == TOOMANYCONST)
           printf("Too many constants in function\n");
        else {
           printf("\n%s\n",function);
           spaces = expression - function;
           for(i=spaces; i>0; --i) printf(" ");
           printf("^\n");
```

(continued from previous page)

```
            switch (SyntaxErr) {
               case MISPLACEDOP:
                  printf("Misplaced operator\n");
                  break;
               case ILLEGALCHAR:
                  printf("Illegal character\n");
                  break;
               case ILLEGALEXP:
                  printf("Illegal exponent\n");
                  break;
               case ILLEGALFUNC:
                  printf("Illegal function\n");
                  break;
               case MISSINGOP:
                  printf("Missing operator\n");
                  break;
               case MISSINGOPRP:
                  printf("Missing operator or right
parenthesis\n");
                  break;
               case MISSINGLP:
                  printf("Missing left parenthesis\n");
                  break;
               case MISSINGRP:
                  printf("Missing right parenthesis\n");
                  break;
               case MISSINGPARM:
                  printf("Missing parameter\n");
                  break;
               case LONEDECIMAL:
                  printf("Lone decimal point\n");
                  break;
               case EXTRADECIMAL:
                  printf("Extra decimal\n");
                  break;
               case EXTRAE:
                  printf("Extra E\n");
                  break;

            } /* switch */

         } /* else */

      } /* if */

      else {
         double x, y, result;

         do {
            printf("\nInput X: ");
            scanf("%lf",&x);
            printf("Input Y: ");
            scanf("%lf",&y);

            result = Evaluate(x,y);

            printf("\nX = %f\n",x);
            printf("Y = %f\n",y);
            printf("Result of %s: %f\n\n", function, result);

            printf("Another calculation? (Y or N) ");
            scanf("%s",answer1);

         } while( (*answer1 == 'y') || (*answer1 == 'Y') );

      } /* else */

      printf("\nAnother function? (Y or N) ");
      scanf("%s",answer2);

   } while( (*answer2 == 'y') || (*answer2 == 'Y') );

} /* main */
```

•AC•

(continued from page 52)

```
  NEXT x
  LINE (128,32)-(464,144),2,bf
  COLOR 1,2
  FOR x=1 TO 10
    IF x=here THEN COLOR 3,2 ELSE COLOR 1,2
    LOCATE 6+x,20:PRINT names$(x)
    LOCATE 6+x,50:PRINT scores(x)
  NEXT x

  COLOR 3,2
  WHILE INKEY$<>""        'empty KB buffer
  WEND
  LOCATE 6+here,18:INPUT "";nam$
  nam$=LEFT$(nam$,20)
  names$(here)=nam$
  LOCATE 6+here,18
  PRINT "                                          ":COLOR 3,2
  LOCATE 6+here,20:PRINT nam$
  LOCATE 6+here,50:PRINT scores(here)
  COLOR 1,0

  OPEN "letters.score" FOR OUTPUT AS #1
  FOR x=1 TO 10
    WRITE #1,names$(x),scores(x)
  NEXT
  CLOSE #1
  GOSUB key.wait
RETURN

key.wait:
  LOCATE 3,28:PRINT "Hit any key to continue."
  WHILE INKEY$=""
  WEND
  LOCATE 3,28:PRINT "                          "
RETURN
```

•AC•

The Fred Fish Public Domain Software Library

The Fred Fish disks are collected by Mr. Fred Fish, a good and active friend of the Amiga.

Fred Fish Disk 179

DietAid — Diet planning aid to allow the user to compile lists of ingredients (recipes) and automatically compute calorie totals, etc. Update FF36. V3.1, binary only. by Terry Gintz

Dmake — Beta release of Matt's version of the UNIX make utility. Features multiple dependancies, wildcard support, and more. Includes source. By Matt Dillion

Excption — Exception is a set of error handling routines that provide a programmer with the ability to easily handle often difficult to implement routines. Routines such as no more memory, file not open, read/write error....etc. V0.6, includes source. By Gerald T Hewes

KickFont — For A-1000 owners, will permanently replace the topaz font on the kickstart disk with a font called "look". Includes a sample in the form of an IFF picture. V3.0, binary only. Also included is Benjamin Fuller's freely redistributable 'SumKick' program. By Greg Browne

Launch — Sample program showing how you can load and execute a program in the workbench environment, then return to the CLI. Includes source. By Peter da Silva

Regexp — A nearly-public-domain reimplementation of the V8 regexp(3) package. Gives C programs the ability to use egrep-style regular expressions, and does it in a much cleaner fashion than the analogous routines in SysV. Includes source. By Henry Spencer

TSnip — Very nice "cut and paste" type utility with lots of uses and functions. Features a pop-up intuition control panel, multiple font and color recognition, clipboard and pipe support and a couple of utility programs. V1.4a, source for support programs only. By John Russell

UnixUtil — A few CLI utilities, including some functionally similiar to the UNIX utilities of the same names. Included are: Wc, Head, Tail, Tee, Detab, Entab, and Trunc. Descriptions are given in the included '.doc' files. By Gary Brant

Fred Fish Disk180

Browser — A programmer's "Workbench". Allows you to easily and conveniently move, copy, rename, and delete files & directories from a CLI environment. Also provides a method to execute either Workbench or CLI programs. V1.6, update to FF134, binary only. By Peter da Silva

GeoTime — A couple of interesting "clock" type programs based on the "Geochron". Observe the earth's shadow scroll across a map or globe in real-time, based on the system clock. V1.0, binary only, shareware. By Mike Smithwick

GPrint — A black & white graphics print utility for Epson compatible printers. Command-line options allow several different print qualities and densities. Includes a couple of sample IFF files for printing. V2.03, binary only, shareware. By Peter Cherna

Jed — A nicely done, intuiton-based editor that is quite user-friendly. Features word-wrap, auto-indent, newcli, alt buffer, split-window, keyboard macro, help, printing, and more. V1.0, binary only, shareware. By Dan Burris

NoVirus — Another Anti-Virus utility. This one features known and new virus detection, view boot block, save and restore bootblocks, several "install" options and more. Written in assembly. V1.56, binary only. By Nic Wilson

RepString — Nice little CLI utility to replace any type of string in any type of file with another string of any type. V1.0, binary only, shareware. By Luciano Bertato

TrekTrivia — Very nice mouse-driven trivia type program for Star Trek fans. Contains 100 questions with additional trivia disks available from the author. Binary only, shareware. By George Broussard

Fred Fish Disk 181

AMXLISP — Amiga-ized version of the XLisp interpreter originally by David Betz. V2.00, includes source. By David Betz; Amiga work by Francois Rouaix

Bally — Amiga port of the former arcade game named Click. Lacks sound effects promised for later updates. V0.1, binary only, shareware. By Oliver Wagner

Tracker — Useful debugging routines similar in function but more versatile to those of "MemTrace" on FF163. Will track and report on calls to AllocMem(), FreeMem() [or lack thereof!] among others. V0.0a (Alpha release). By Karl Lehenbauer

Fred Fish Disk 182

AMC — "Amiga Message Center". Scrolls a message from a text file across the screen on a colorful background. Similar to the "greetings" programs developed by European Amiga enthusiasts. V1.0, binary only. By Foster Hall

Edimap — A keymap editor. Allows you to read in an existing keymap file, modify it to suit your needs, and save it as a ready-to-use keymap. V1.0, includes source. Author: Gilles Gamesh

HR136 — An IFF file containing a chart showing every possible mixture of the sixteen basic palette colors. Also included are optimized and monochrome palettes along with several tips and techniques for using them with various paint programs.By Dick Bourne

Iconmerger — Intuition-based program to take any two brush files and merge them into an alternate-image type icon. V2.0, binary only. By Terry Gintz

Sam — Another IFF sound player with several command-line options. Includes several samples. V1.0, binary only. By Nic Wilson

SetFont — Allows you to change the system font with various command-line options. Cleans up all known bugs in FF75. V2.5, includes source in C++. By Dave Haynie

Fred Fish Disk 183

FixFd — A utility for Amiga assembly programmers. FixFd will read a '.FD' file and output a file that can be 'INCLUDE'ed rather than having to link with the colossal 'Amiga.Lib'. V1.0, includes source in assembly. By Peter Wyspianski

Mklib — Another example of building a shared library that evolved from "Elib" FF87. Also included is a library, Edlib, which contains several functions not included in the Manx standard libraries. Includes source. By Edwin Hoogerbeets with C-functions from several different authors

PCQ — A subset implementation of a freely-redistributable Pascal compiler. Supports include files, external references, records, enumerated types, pointers, arrays, strings and more. Presently does not support range types, the 'with' statement or sets. V1.0, includes source and sample programs. By Patrick Quaid

Fred Fish Disk 184

BI — A small brush to to C-code image converter, intended to be used from CLI. V1.0, binary only. By Terry Gintz

CardMaker — A programmer's aid for creating card image data that can be used in any card game that uses the standard 52 card deck. V1.0, binary only. By Terry Gintz

DPS — Demo version of a program that will allow you to take any IFF file and save it as a totally self-contained executable file, without the need for any IFF-viewers. V1.0, binary only. By Foster Hall

MouseUtil — Intuition based program to allow you to change your mouse speed without having to go through preferences. V1.1, includes assembly source. By Luciano Bertato

Print — Small print utility designed to replace the "copy <filename> to prt:" command. Opens a window displaying the filename being printed, length, and a status bar showing percent completed. Also includes an abort gadget. V1.0, binary only. By Luciano Bertato

VacBench — This amusing little screen hack will "clean up" your WorkBench screen for you when it gets too cluttered! Binary only. By Randy Jouett

World — A text adventure game similar to the Infocom adventures of Planetfall and Starcross. Quite large with a tremendous variety of responses. V1.02, includes source. By Doug Mcdonald, Amiga port by Eric Kennedy

Fred Fish Disk 185

Commodore IFF — This is a copy of the official November 1988 Commodore IFF disk. All the files in the "documents" directory are in zoo file "documents.zoo"

Fred Fish Disk 186

A68k — A 68000 assembler originally written in Modula-2 in 1985 and converted to C by Charlie Gibb in 1987. Has been converted to accept metacomco-compatible assembler source code and to generate Amiga objects. Includes source. This is V2.42, an update to FF110. By Brian Anderson; C translation and Amiga work by Charlie Gibb

Cards'O'Rama — A simple game that let's you push your memory. It is played with a deck of 32 cards, grouped in 16 pairs. The cards are shuffled and then displayed at the beginning of each game. Your goal is to pick up as many pairs as you can, until there are no cards left on the screen. V1.0, includes source. By Werther Pirani

Qt2 — A cute program that gives the time the way many people actually do, I.E. "it's nearly ten to five". Includes source in assembly. By Charlie Gibb

SimCPM — A CP/M simulator for the Amiga. Simulates an 8080 along with H19 terminal emulation. Includes source. This is V2.3, an update to FF109. By Jim Cathey; Amiga port by Charlie Gibbs and Willi Kusche

Fred Fish Disk 187

Diskpert — A disk benchmark program which runs on both Unix and the Amiga. This is an update to FF48, with bug fixes and more reliable measurements of the faster read and write speeds available under the new Fast File System. By Rick Spanbauer, enhancements by Joanne Dow

HackLite — This is the latest version the Amiga port of Hack, with lots of Amiga specific enhancements and neat graphics. Now includes an easy to use installation program. This is HackLite V1.0.0, binary only. By Software Distillery

Mackie — A versatile cli/macro-key initiator based on POPCLI with a unique method of "screen-blanking". I won't say more, just try it! V 1.13, includes source. This is an update to FF161. By Tomas Rokicki

SetCPU — A program designed to allow the user to detect and modify various parameters related to 32 bit CPUs. Includes commands to enable or disable the text/data caches, switch on or off the 030 burst cache line fill request, use the MMU to run a ROM image from 32-bit memory, and to report various parameters when called from a script. V1.4, includes source. By Dave Haynie

Fred Fish Disk 188

BootIntro — This program creates a small intro on the bootblock of any disk, which will appear after you insert the disk for booting. The headline can be up to 20 characters. The scrolling text portion can be up to 225 characters. V1.0, binary only. By Roger Fischlin

DiffDir — Diffdir compares the contents of two directories, reporting on differences such as files present in only one directory, different modification dates, file flags, sizes, comments, etc. V1.0, includes source. By Mark Rinfret

ExecDis — A disassembler comment generator program for the 1.2 Kickstart ROM exec library image. Generates a commented disassembly of the exec library. V1.0, binary only. By Markus Wandel

FastGro — A fractal program, simulating Diffusion-Limited Aggregation (DLA) as described in the December 1988 Scientific American in the Computer Recreations column. This program is about an order of magnitude faster than the "SLO GRO" program described in Scientific American. V 1.0, includes source. By Doug Houck

FracGen — A fractal generator program that generates fractal pictures from "seeds" that you create. This is unlike any of the other "fractal generators" I've seen. It can be used to load and display previously created fractal pictures, modify existing fractals, or create your own fractals. V 1.23, binary only, update to FF142. By Doug Houck

MemoryClock — A clock program that shows the amount of free fast ram, free chip ram, as well as the time and date. Includes source in assembly code. By Roger Fischlin

MinRexx — A simple ARexx interface which can be easily patched into almost any program. Includes as an example the freedraw program from FF1. Includes source. By Tomas Rokicki

Null — A new dos device that behaves like "NIL:" but unlike "NIL:", it is a real handler. This makes it useful in lots of situations where "NIL:" cannot be used. V 0.0. includes source. By Gunnar Nordmark

TextDisplay — A text display program, like "more" or "less", but about half the size and handles all screen formats (pal/ntsc, interlace/non-interlace, etc). V 1.1, binary only. By Roger Fischlin

Fred Fish Disk 189

Mackie — A versatile cli/macro-key initiator based on POPCLI wit a unique method of "screen-blanking". I won't say more, just try it! Version 1.20, includes source. update of FF187. Author: Tomas Rokicki

NetHack — This is part 1 of a two part distribution of NetHack, which was too large to fit on a single disk, even when zoo'd. Part 2 is on disk 190. Both parts, along with zoo to unpack them, are required to use or rebuild NetHack. V 2.3. Includes source. Author: Various; Amiga work by Olaf Seibert

Uedit — V 2.4g shareware editor. Has learn mode, a command language, menu customization, and other user configurability and customizability features. Binary only, shareware, Update to FF 173 Author: Rick Stiles

Fred Fish Disk 190

GaryIcons — A collection of more interesting and useful icons. Author: Gary Roseman

LBM2Image — Takes an IFF picture and generates a C source module which can be compiled and linked with your program to display the picture with the intuition DrawImage function. Binary only. by: Denis Green

NetHack — This is part 2 of a two part distribution of NetHack, which was too large to fit on a single disk, even when zoo'd. Part 1 is on disk 189. Both parts, along with zoo to unpack them, are required to use or rebuild NetHack. V 2.3 Includes source. Author: Various; Amiga work by Olaf Seibert

Fred Fish Disk 191

BlitLab — Blitlab is a program which lets you experiment with the blitter, to your hearts content, in relative safety. It opens a workbench window with gadgets for all the registers of the blitter, and allows you to manipulate individual registers and perform blits on a magnified bitmap. V 1.4, an update to FF84. Includes source. Author: Tomas Rokicki

Blk — A requester making tool employing various recursive algorithms including a recursive parser. It takes input text files and converts them to C-source for including as requester declarations. Update to FF152, with many enhancements. Includes source. Author: Stuart Ferguson

FileBootBlock — This simple little program reads blocks 0 and 1 of a bootable disk and saves them as a program file that can be run (heaven forbid) or disassembled by programs like DIS or DSM. Includes source in assembly code. by: John Veldthuis

Spell — A port of a Unix version of a freely distributable screen oriented, interactive, spelling checker. Update to FF54, with enhancements by Tomas Rokicki. V2.0.02, includes source. Author: Pace Willisson; enhancements by Tomas Rokicki

Pz15 — Computer version of those cheap plastic puzzles with 15 white tiles numbered 1 through 15 and an empty square in a 4 by 4 arrangement. This one is more challenging since you can't solve it by just prying out the pieces. Includes source. Author: Mike Hall

Fred Fish Disk 192

Eval — This package allows you to manipulate expressions. Currently its two main functions are evaluation and differentiation. It also does some basic simplifications (based on pattern matching) to make the result of a differentiation more presentable. Includes source. Author: David Gay

PacMan87 — This is a nice little "pacman like" game with some new features like fire pits, stabbing knives, electric arcs and flame throwers, that must be avoided. Has three levels of difficulty, easy, medium, and hard. Sounds can be toggled on or off. Keeps a record of the top ten scores. Shareware, binary only. Author: Steve Jacobs and Jim Boyd

ReSourceDemo — A demo version of ReSource, an interactive disassembler for the Amiga. This is a complete version except that the "save" features have been disabled. V0.36, binary only. by: Glen McDiarmid

Fred Fish Disk 193

KeyMapEd — Allows you to change the KeyMaps used with SetMap This is a full featured editor providing support for normal, string and dead keys. The keyboard represented is from an A2000/A500 but it is fully compatible with A1000 keyboards. V1.02, includes source Author: Tim Friest

Zc — This is a modified version of the Sozobon C compiler fromFF171. It has been modified to generate code compatible with the A68k assembler from FF186 and a new frontend control program makes it easy to use like the UNIX "cc" frontend. V1.01, includes source. by: Johann Ruegg; Amiga work by Joe Montgomery

Fred Fish Disk 194

Mona — A single player dungeon simulation. The object of the game is to defeat the Balrog, which lurks in the deepest levels of the dungeon. You begin at the town level above the dungeon, where you may acquire supplies, weapons, armor, and magical devices by bartering with various shop owners, before descending into the dungeon to do battle. Amiga enhancements include pull down menus, graphics mode, pickup mode, a continuous move mode, a real time mode, a message wait time mode, as well as other modifications to improve overall playability and to take advantage of the unique features of the Amiga. V3.0, binary only, requires at least 1Mb of memory. Author: Robert Alan Koeneke and others. Amiga version by Richard Henderson and others.

Fred Fish Disk 195

MicroEMACS — Version 3.10 of Daniel Lawrence's variant of Dave Conroy's microemacs. This is an update to the version released on disk 119. New features include multiple marks, more function key support, a better crypt algorithm, and end-of-word command, a command line switch for setting environment variables, new hooks for macros, a command to strip trailing whitespace, international-zation features like foreign language message support, horizontal window scrolling, much faster search algorithm, Amiga intuition support, and more. Includes source and extensive online documentation. Author: Dave Conroy, MANY enhancements by Daniel Lawrence

Fred Fish Disk 196

HamPics — These are some of the most stunning digitized pictures yet for the Amiga. They were scanned at a resolution of 4096 by 2800 pixels, 36-bits per pixel, on an Eikonix 1435 slide scanner, cropped, gamma corrected, scaled, and converted to Amiga IFF HAM files. They are displayed with a special ILBM loader that handles overscan HAM images. Includes source for the display program. Author: Jonathan Hue

Fred Fish Disk 197

CTags — Create a tags file from the specified C, Pascal, Fortran, YACC, lex, or lisp sources. A tags file can be used by a cooperating editor to quickly locate specified objects in a program's source code. Berkeley V4.7, includes source. Author: Ken Arnold, Jim Kleckner, and Bill Joy Ported to Amiga by G. R. (Fred) Walter

Find — Find is a utility which searches for files that satisfy a given boolean expression of attributes, starting from a root pathname and searching recursively down through the hierarchy of the file system. Very much like the Unix find program. V1.2, includes source. Update to FF134 Author: Rodney Lewis

FixHunk — A program to modify executable files to allow them to run in external memory. It forces all DATA and BSS hunks in the file to be loaded into CHIP memory. CODE hunks will still load into FAST ram if available. New features include an interactive mode to select where each DATA or BSS hunk will load into memory, support for overlays, support for AC BASIC compiled programs, and support for new hunk types as used by "blink". V2.1, binary only. Update to FF36. Author: D.J. James

Nro — Another roff style text formatter. This is version 1.5, an update to the version released on disk 79. New features include generation of ANSI/ISO codes for bold, italics, and underline, more than one formatting command on a line, longer macro names, and many more formatting commands. Includes source. Author: Unknown, posted to usenet by Alan Vymetalik Many enhancements by Olaf Seibert

Stevie — A public domain clone of the UNIX 'vi' editor. Supports window-sizing, arrow keys, and the help key. V3.35a, includes source. Update to FF166. Author: Various, Amiga work by G. R. (Fred) Walter

Fred Fish Disk 198

Charon — Charon is Bradley's entry for the First Annual Badge Killer Demo Contest. The text of the demo was written by Lord Dunsany (long before the Amiga). Bradley created the illustrations and animation. The sound track is a traditional Scottish tune "The Arran Boat". by: Lord Dunsany (1915), Bradley Schenck (1988)

Fred Fish Disk 199

ASimplex — An implementation of the Simplex algorithm for solving linear programs. It uses the standardized MPSX-format for input data files. V1.2, includes source. Author: Stefan Forster

Csh — V3.02a of a csh like shell derived from Matt Dillon's shell, V2.07. Includes many new or improved commands, some bug fixes, etc. Includes source. Author: Matt Dillon, Steve Drew, Carlo Borreo, Cesare Dieni

MIDIsoft — A program to transfer sound samples between the Amiga and a Roland S-220. V1.0, binary only. Author: Dieter Bruns

Pyro — A screen blanking program that goes beyond the normal blanking process. When there are no input events, pyro takes over and starts a little fireworks display in color. V1.1, binary only. Author: Steve Jacobs and Jim Boyd

SnipDemo — Demo version 1.23 of signal processing program sold by Digital Dynamics. Binary only. Author: John Hodgson

Viewer — A very small program for displaying IFF pictures of any resolution. This one is written in assembly code and is only 988 bytes long. Binary only. Author: Mike McKittrick

Fred Fish Disk 200

NotBoingAgain — Dr. Gandalf's entry for the First Annual Badge Killer Demo Contest. It is an interlaced HAM animation with nicely integrated sound effects. It is a great visual pun on the original Boing demo, but to say anymore would ruin the effect. Binary only, requires 1 Mb of memory. Author: Dr. Gandalf (Eric J. Fleischer, MD)

Tank — This is Vincent's entry for the First Annual Badge Killer Demo Contest. It is an animation of a "fishtank simulator", with sound effects and a cute twist. Binary only. Author: Vincent H. Lee

For PDS orders, please use form on page CIII

Visa and MasterCard is available on orders of $20.00 or more.

Fred Fish Disk 201

Draco Update to Chris Gray's Draco distribution for the Amiga.Enhancements include support for floating point, register variables, more optimization, improved call/returnstandard, etc. V1.2, an update to FF76. Requires documentation from FF77 to complete the distribution kit. Binary only. Author: Chris Gray

DropCloth DropCloth lets you place a pattern, a 2 bitplane IFF image or a combination of a pattern and image, into the WorkBench backdrop. This is version 2.4, an update to version 2.2 on disk 128. Shareware, binary only. Author: Eric Lavitsky

Fred Fish Disk 202

SlavicFonts A whole bunch of new fonts from Robin LaPasha: Version 1.0. Author: Robin LaPasha

Vlt VLT is both a VT100 emulator and a Tektronix (4014 plus subset of 4105) emulator, currently in use at SLAC (Stanford Linear Accelerator Center). Although the VT100 part was originally based on Dave Wecker et al.'s VT100, many enhancements were made. The program requires ARP, and it has an ARexx port. XMODEM 1K/CRC and Kermit protocol support also included. Version 3.656, binary only. Author: Willy Langeveld

Fred Fish Disk 203

Examples Assembly and C code examples, including some old favorites (like speechtoy and yachtc3) downcoded to assembly language. Includes a replacement for the official audio device, an example of creating a subtask, a rewrite in assembly of R. J. Mical's file requester, an example of installing a custom input handler ahead of intuition, and more. Author: Jim Fiore & Jeff Glatt

GurusGuide The source files for all examples published in the "Guru's Guide, Meditation #1: Interrupts" by Carl Sassenrath, the architect of the Amiga's low-level multitasking operating system and designer of Exec. Author: Carl Sassenrath

Isam A library of routines to access relational data base systems using the Index Sequential Access Method (ISAM). This is beta version 0.9, binary only. Author: Kai Oliver Ploog

Fred Fish Disk 204

FileReq A simple file requestor, written as an exercise by the author to see how easy it would be (it wasn't). Includes source. Author: Jonathan Potter

GnuGrep The grep program from the GNU project. Replaces grep fgrep, egrep, and bmgrep. Currently does not expand Amiga style wildcards, so if you wish to scan multiple files you will need to use it with a shell that does this for you. Version 1.3, includes source. Author: Mike Haertel, James Woods, Arthur Olson, Richard Stallman, Doug Gwyn, Scott Anderson, Henry Spencer

HAMCu Installs a custom copper list for the current active view (usually workbench) that contains all the colours from 0x000 to 0xfff. A neat effect and an easy way to show off the color capabilities of the Amiga. Includes source. Author: Jonathan Potter

Image-Ed An shareware icon editor submitted by the author for inclusion in the library. Suggested shareware donation of $20. Version 1.8, binary only. Author: Jonathan Potter

JPClock A short clock program that is just packed with features. Includes source. Author: Jonathan Potter

MouseBounce A short hack/game that makes your mouse pointer bounce around the screen. The object is to close the MouseBounce window and exit the game. Each time you click the mouse button, the pointer speeds up. Includes source. by: Jonathan Potter

PopDir A small utility which "pops open" to help you look at the contents of a particular directory on demand. Version 1.4, includes source. by: Jonathan Potter

PopInfo A small utility which "pops open" to give you information about the status of your devices & memory. V 2.9, includes source. by: J. Potter

Teacher Teacher is a short, simple hack. I won't spoil the fun by telling you what it does. Includes source. Author: Jonathan Potter

Fred Fish Disk 205

Bally Amiga port of the former arcade game named Click. This version now has sound effects. Version II, an update to FF181. Binary only, shareware. by: Oliver Wagner

BattleForce A nicely done shareware game, submitted by the author, that simulates combat between two or more giant, robot-like machines. Binary only, version 3.01. Author: Ralph Reed

Chess A port of a chess game posted to Usenet. This is an update to the version first included on disk 96. It has been upgraded to use an Amiga Intuition interface. Version 2.0, binary only. Author: John Stanback; ported to Amiga by Bob Leivian Version 2.0 upgrades by Alfred Kaufmann

Fred Fish Disk 206

Brownian A demo based on both fractal theory and brownian motion. Includes source. Author: John M. Olsen

Hawk A stereo image of a hawk. Requires red/green stereo glasses to view. Author: Unknown (no documentation included)

MemFlick Treats all the memory in your Amiga like it was part of a bitplane inside a graphics display. Provides sort of a graphical picture of your memory usage. Binary only. by : Jim Webster

PeX A demo of the various graphics capabilities of the Amiga. Author: Unknown (no documentation included) PictureGarden Another demo, apparently in compiled basic. Author: Unknown (no documentation included)

StereoDemo A demo of stereoscopic graphics, written in assembly language. Requires red/green stereo glasses to view. Includes sources. Author: David M. McKinstry

Triple 3 demos of some of the Amiga's graphics and sound capabilities. Binary only. by: Tomas Rokicki

Fred Fish Disk 207

Coyote Gene's entry to the 1988 Badge Killer Demo contest. A very cute (and large) animation. Requires about 1900 blocks of disk space, so it is distributed in "arc format". Author: Gene Brawn

Fred Fish Disk 208

AsteroidField This is Michael's entry for the 1988 Badge Killer Demo Contest. It is a large animation of a spacecraft flying madly through an asteroid field (chased by unseen foes) that includes a couple of near misses. Author: Michael Powell

Fred Fish Disk 209

Bowl This is Vern's entry for the 1988 Badge Killer Demo Contest. It is a Sculpt-Animate animation that shows three colored balls flying in circles above a mirrored bowl. Rendering the animation took about 2 weeks. Distributed in zoo format because of its size (zoo program included for easy unpacking). Author: Vern Staats

Dps A program designed to work with the PrintScript program, a commercial PostScript interpreter for the Amiga, to provide a page previewer. V 1.1 and includes source. by: Allen Norskog

Fred Fish Disk 210

Calc A very nicely done scientific/programmer/plotter calculator. The scientific portion has most of the operations found on the more popular handhelds. The programmer portion has all the special hex/binary/decimal conversions as well as register operations like ASL, ROL, LSL, AND, OR, XOR, etc. The plotter portion will plot equations. Other features include 26 memories, full mouse or keyboard operation, pull-down menus, and iconization. V 3.0, binary only. by: Jimmy Yang

LabelPrint A program that allows you to easily print labels for your disks. Version 1.9, shareware, binary only (source available from author). Author: Andreas Krebs

NuHand An animation of a hand with fingernails scrapeing on a desktop, including sound effects. This is Bryan's entry for the 1988 Badge Killer Demo Contest. Binary only by: Bryan Carey Gallivan

Fred Fidh Disk 211

AmigaWave This is Allen's entry to the 1988 Badge Killer Demo Contest. It is an animation with sound effects by: Allen Hastings

Esperanto A keymap modification to usa1 which, in conjunction with the supplied slate.font, will allow one to type in Esperanto and Welsh, in any program that will use keymaps & fonts by: Glyn Gowing

Image-Ed An shareware icon editor submitted by the author for inclusion in the library. Suggested shareware donation of $20. V1.9, binary only. Fixes a serious bug in V1.8 on FF204. by: J. Potter

SignFont A keymap and font that will allow the user to be able to type in American Sign Language, provided that one knows the font. Author: Glyn Gowing

VirusControl A new virus detection and control program that checks disks during insertion, protects from link viruses, shows bootblock on a screen, periodically checks system vectors, controls access to files with a requester, etc. V1.3, includes full assembly language source code. Author: Pius Nippgen

Fred Fish Disk 212

Alice This animation is Carey's entry to the 1988 Badge Killer Demo Contest. Author: Carey T. Pelto

DiskSalv A disk recovery program for all Amiga file system devices that use either the Amiga Standard File System or the Amiga Fast File System. Disksalv creates a new filesystem structure on another device, with as much data salvaged from the original device as possible. Update to FF177. Binary only. Author: Dave Haynie

DogsWorld This animation is Charles' entry to the 1988 Badge Killer Demo Contest. by: Charles Voner

Fred Fish Disk 213

Cucug This animation of the Champaign-Urbana Commodore Users Group logo was submitted to the 1988 Badge Killer Demo Contest by Ed Serbe. by: Ed Serbe

Icons Almost 300 icons in eight (!) colors. Uses a special program to get an eight color workbench to display these icons, which were made with DPaintII and IconGen. Most icons are miniatures of the main screen of their corresponding programs, or the picture they show, made with "iconize" and "recolor" from FF85. by: Wolf-Peter Dehnick

Fred Fish Disk 214

ArcPrep ArcPrep prepares files and/or directories for archival with arc or any other program that can't scan through different directories and/or handle long filenames. V2.1, includes source. Author: Garry Glendown

MandelVroom A Mandelbrot/Julia-curve generating program that features five numerical generators (integer, ffp, ieee, 020, and 020/881) in hand-crafted assembly for maximum speed, online mouse selectable help for all functions, generation of multiple pictures simultaneously, a sophisticated user interface with shaded gadgets, etc. Some of the other features include zoom, magnify, color-cycling, contouring, auto-contouring, histogram, statistics, presets, extra-halfbrite support, overscan, orbits, pan mode, and more. Requires 1Mb or more of memory. This is the source to V2.0, an update to FF78. A compiled binary, along with help files and example images, can be found on FF215. Author: Kevin Clague

MemDiag A memory diagnostic program to identify addresses which produce memory errors, and a memory quarantine program which removes such defective addresses from the system's free memory list, until such time as the hardware errors can be corrected. Version 1.1, includes source. by: Fabbian Dufoe

RunBack Another step in the evolution of Rob Peck's RunBackGround program, from disks 73 and 152. Allows you to start a new CLI program and run it in the background, then closes the new CLI. This version has been enhanced to use the NULL: device by Gunnar Nordmark (included), which is a "real" device, so it solves problems with previous versions of RunBack which used the Nil: "fake" device, causing many crashes. Includes source. Author: Rob Peck, Daniel Barrett, Tim Maffett

SmartIcon This shareware program, submitted by the author, is an Intuition objects iconifier. Version 1.0 is limited to iconifying windows, which is still very handy. It adds a new "iconify gadget" to each window, that when clicked on, iconifies the window into an icon in the ram: disk. This is the same version as released on FF134, but now includes the source code. Author: Gauthier Groult

Fred Fish Disk 215

MandelVroom A Mandelbrot/Julia-curve generating program that features five numerical generators (integer, ffp, ieee, 020, and 020/881) in hand-crafted assembly for maximum speed, online mouse selectable help for all functions, generation of multiple pictures simultaneously, a sophisticated user interface with shaded gadgets, etc. Some of the other features include zoom, magnify, color-cycling, contouring, auto-contouring, histogram, statistics, presets, extra-halfbrite support, overscan, orbits, pan mode, and more. Requires 1Mb or more of memory. This is V2.0, an update to FF78. Supports window-sizing, arrow keys, and the help key. V3.6, includes source. This is an update to V3.35a on FF197. Author: Various, Amiga work by G. R. (Fred) Walter

Fred Fish Disk 218

EdLib A library of additional functions for Manx. This is V1.1, an update to V1.0 from FF183. Includes source. Author: Edwin Hoogerbeets with C-functions from several different authors

Mandel Another mandelbrot generator program, with bits and pieces of code from C. Heath and R.J. Mical. This is V1.3, an update to FF111. New features and improvements include an ARexx interface, coordinates in sight, more state info saved with a picture, batch files, programmable functions, and more plotting options. Includes source. Author: Olaf Seibert

Maze A program that lets you build mazes and then solve them. Mazes can be trivial one level mazes to very difficult three level mazes. Version 1.2, includes source. Author: Todd Lewis

PcPatch Patches for PCCopy and PCFormat from the EXTRAS disk, to allow reading, writing, and formatting of any kind of MS-Dos style disks, including 720K 3.5" diskettes. Binary only. This is an update to the version on disk 163. Author: Werner Guenther

Scanner Scanner makes commented C code of all intuition structures in memory. The structures will receive correct pointers towards each other. Scanner starts looking at IntuitionBase, and follows all pointers, storing them in memory. When finished, it writes all the structures to the standard output. Version 1.0, includes source. Author: Stefan Parmark

Worm An Amiga implementation of the classic "worms" program, based on an article in the Dec 1987 issue of Scientific American. You can specify the size and length of the worms, and the number of worms. Includes source. Author: Brad Taylor, Amiga port by Chuck McManis

Fred Fish Disk 219

DeepSky A database containing information on 10,368 non-stellar objects, 600 color contrasting easily resolved double stars, 70 stars for setting circles, and misc white dwarfs, red stars, binaries, etc. The database is distributed in zoo format, and is about 1.2 Mb after extraction. V 5.0. Author: Saguaro Astronomy Club

Mv A Unix style mv/cp/rm program that moves, copies, or removes files. Includes interactive mode, recursive mode, and force quiet mode. Copies file permissions, dates, and comments, supports arp style wildcards, supports moves across volumes, honors the delete bit. V 1.1, includes source. Author: Edwin Hoogerbeets

Fred Fish Disk 220

DNet A link protocol that provides essentially an unlimited number of reliable connections between processes on two machines, where each end of the link can be either an Amiga or a Unix (BSD4.3) machine. Works on the Amiga with any EXEC device that looks like the serial device. Works on UNIX with tty and socket devices. Achieves better than 95% average throughput on file transfers. This is V2.0, an update to FF145. Includes sources for both the Amiga and Unix versions. Author: Matt Dillon

Fred Fish Disk 221

AllocMaster Allocmaster is a program inspired by Nick Sullivan's "Reserve" article in Amiga Transactor, for controlling the amount of both Chip and Fast memory available to the rest of the system. It is very useful for testing applications in low-memory situations. It also has a snapshot feature to report differences in available memory before and after running an application. V 1.17, binary only. Author: John Gerlach Jr.

ANSIEd Demo version of an ANSI screen file editor. It allows you to easily create and modify a screen of ANSI-style text/graphics on the Amiga. The standard ANSI color set (red, green, yellow, blue, magenta, cyan, white) and text styles (plain, boldface, underlined, italic) are provided, along with some simple editing and drawing functions. This demo version has the save features disabled. V1.2.0aD, binary only. Author: Gregory Epley

BallyII Amiga port of the former arcade game named Click. This version adds a "cheat" mode and fixes some minor bugs. VII+, an update to FF205. Binary only, shareware.Author: Oliver Wagner

DFrame A utility that helps you to create animated bobs. It installs itself in DPaint II, after which you can draw each bob in DPaint II within its own frame and check the animation by calling DFrame from within DPaint. V1.02, binary only.Author: Jan Buitenhuis

IFFM2 Demo version of an IFF support module for Interface Technologies M2Amiga Modula-2 system. Includes a version of ViewILBM (with source) that uses the IFF support routines. V1.0.0D, binary only. Author: Gregory Epley

Steinschlag A tetris like game (Steinschlag means "Falling Rock") submitted by the author. V1.5, binary only. Author: Peter Handel

Fred Fish Disk 222

MemGauge A tool to display the current memory usage, very much like the usage bar Workbench displays in root directories. V1.4, includes source. Author: Olaf 'Olsen' Barthel

Mischief This little program is in the long tradition of "display hacks". It uses the input.device to perform various acts of mischief. Includes source. Author: Olaf 'Olsen' Barthel

Plplot A library of C functions useful for scientific plotting on the Amiga. The library is Lattice C compatible. Contour plotting, three dimensional plotting, axis redefinition, log-log plotting and multiple subpages are a few of Plplot's features. The plots can be displayed on a monitor or sent to a graphics file for subsequent printing. V1.00, includes source. Author: Tony Richardson

Fred Fish Disk 223

Csh V3.03a of a csh like shell derived from Matt Dillon's shell, version 2.07. This is an update to FF199. Includes a couple of new filter commands, new dir option, new editing options, sourcing of a standard startup file, and some bug fixes. Includes source. Author: Matt Dillon, Steve Drew, Carlo Borreo, Cesare Dieni

FixDisk A program to recover as much as possible from a defective disk. It can sometimes recover damaged (unreadable) tracks, check file integrity, check the directory structure, undelete files, copy or show files, fix corrupted directory pointers, etc. Full intuition interface. Version 1.0, binary only. Author: Werner Guenther

GravSim A program to animate up to 6 planetary masses, all of which exert a mutual gravitational force on each other. The planetary masses can be placed anywhere on the screen, and their mass and initial velocity can be determined by the user. The program then steps the animation through time, plotting and displaying the new position in the trajectory of each mass. V1.50, includes source. Author: Richard Frost

Iff2Sun A small utility for those of you who may have access to a Sun workstation. Takes an Amiga IFF file and converts it to a Sun rasterfile format. Update to FF174, with better parsing, support for HAM mode, and some bug fixes. Source only, as the program needs to be re-compiled and run from a Sun. Authors: Steve Berry, Mark Thompson

IFFtoSUN This program takes a standard IFF format image and translates it into a SUN rasterfile format, like the Iff2Sun program also on this disk. However, this one runs on the Amiga. V1.31, includes source. Author: Richard Frost

Pacer A pacman clone with sound and a game screen editor. V1.0, shareware, binary only. Author: Dirk Hoffman

PopInfo A small utility which "pops open" to give you information about the status of your devices and memory. V3.0, an update to FF204. Includes source. Author: Jonathan Potter

SetCPU A program designed to allow the user to detect and modify various parameters related to 32 bit CPUs. Includes commands to enable or disable the text/data caches, switch on or off the '030 burst cache line fill request, use the MMU to run a ROM image from 32-bit memory, and to report various parameters when called from a script. V1.5, an update to FF187. Includes source. Author: Dave Haynie

Fred Fish Disk 224

CLImax For all those people who wish that their CLI windows had 25 lines of 80 characters just like an old fashioned non- windowing computer, the answer is here. CLImax creates a borderless backdrop CLI window on a custom screen. Also thrown in is MoveSys, which reassigns SYS:, C:, S:, L:, DEVS:, LIBS:, and FONTS: to a new volume with one simple "pure" command. Includes source. Author: Paul Kienitz

KickMem A program for A1000 hardware hackers that have done the Amazing Computing 512K upgrade. KickMem will patch your 1.2 or 1.3 kickstart disk to perform addmem during kickstart. This allows warm boot surviveability of ram disk devices and eliminates addmem commands from your startup sequence. V2.0, includes source. Author: Dave Williams

MoreIsBetter These two hacks make MORE more useful. One is called V; it's a small "pure" CLI command that acts as a front end for More, causing it to create its own window. Make V and More both resident! The other is Fenestrate, which surgically alters the CON: window spec inside More enabling it to, for instance, use ConMan features to create a borderless window on the topmost screen (very useful with CLImax). Includes source. Author: Paul Kienitz

PetersQuest This cute game has you, the intrepid Peter, following a trail of hearts through a world of 20 levels, riddled with porcupines and other hazards, to rescue Daphne, the love of your life that has been kidnapped by the evil Brutus. Version 1.0, binary only. Author: David Meny

Who A rewrite of "who", from FF79, which gives substantially more elaborate information about the tasks currently running (or waiting) on your Amiga. Includes source. Author: George Musser, rewrite by Paul Kienitz

Xebec A couple of hacks to make life easier for those who have Xebec hard disks. One makes it more possible to Mount a Xebec hard disk with the Fast File System, the other is a compact head parking program. Includes source. Author: Paul Kienitz

Fred Fish Disk 225

AmigaTCP The KA9Q Internet Software Package. The package supports IP, ICMP, TCP, UDP, and ARP as basic services, and implements the FTP, Telnet, and SMTP protocols as applications. It runs on IBM PC and clones, the Apple Macintosh, and the Amiga. Includes source. Author: Bdale Garbee, Phil Karn, Brian Lloyd

MyMenu MyMenu allows you to add your own menus to the WorkBench menu strip, to run commonly used commands. MyMenu will allow you to execute both CLI and WorkBench programs, and is configured with a normal text file. Includes source. Author: Darin Johnson

Fred Fsih Disk 226

Vlt VLT is both a VT100 emulator and a Tektronix (4014 plus subset of 4105) emulator, currently in use at SLAC (Stanford Linear Accelerator

Source is available on FF214. by: Kevin Clague

Fred Fish Disk 216

BackDrop Backdrop allows you to define a pattern which will then be displayed on the workbench screen in the normally empty area behind all the windows. Similar in concept to DropCloth, but this one does not require workbench to be loaded (and does not cohabit very well with workbench). Includes source. Author: Eddy Carroll

C64Emul An April Fools spoof that turns your Amiga into a C64, or at least makes it look that way. Includes source. Author: Eddy Carroll

Cloud A program that generates and displays fractal surfaces that look remarkably like clouds. Based on ideas from the book "Fractals" by Jens Feder. Binary only. Author: Mike Hall

PrtSpool A DOS handler, a print program, and a control program that implement a print spooling system. Like PRT:, the DOS handler waits for stuff to be sent to it to be printed. The print program does line numbering and page headers. The control program handles administrative functions. Binary only. Author: Daniel Barrens

VirusX Version 3.20 of the popular virus detection/ vaccination program. Features a test for 8 new viruses since the 3.10 version on FF175. Includes source. Author: Steve Tibbett

Wanderer A neat little game with graphics and sound, ported from the Unix version, originally written on a Sun workstation. The idea for Wanderer came from games such as Boulderdash, Xor, and the Repton games from Superior Software. Includes a builtin editor for extending the game by adding additional screens. V2.2, includes source. Author: Steven Shipway and others. Amiga port by Alan Bland

Fred Fish Disk 217

AntiCBS An animation cooked up by Leo in protest of CBS's coverage of the Hacker's Conference in Oct 88. After reading the transcript I was angered enough to feel this needed widespread distribution, even though it is quite old. by: Leo 'Bols Ewhac' Schwab

Echo A small replacement for the AmigaDOS echo that will do some special functions, such as clear the screen, delete to bottom of screen, scroll the screen, place the cursor at a particular location, and set the text style and/or color. Includes source. Author: Garry Glendown

InstallBeep This program replaces the DisplayBeep function so that an IFF 8SVX sound is played instead of the screen flashing. The PlayBeep function runs as a task in the background and runs asynchronously so the length of the sound does not slow anything down. Includes a couple of sample sound files. Version 1.1, binary only. Author: Tim Friest and Don Withey

SnipIt An input handler wedge which allows you to clip text from any window and then paste that text anywhere, as though you had typed it on the keyboard. You mark the text you want to "snip" using the mouse, and then use the mouse to "paste" the last snipped text into the active window, requester, or anywhere. Version 1.2, includes source. Author: Scott Evernden

SonixPeek A utility to let you list all the instruments used by one or more Aegis Sonix score files. It can scan individual files, or search one or more directories, checking all score files in each directory. The output is a list of all the instruments you need to have present in order to be able to play the indicated score files. Includes source. Author: Eddy Carroll

Stevie A public domain clone of the UNIX 'vi' editor, were made. The program requires ARP, and it has an ARexx port. XMODEM 1K/CRC and Kermit protocol support also included. V4.036, with many enhancements over the previous version, 3.656, on FF202. New features include support for other serial ports, external file transfer protocols, and "chat" mode. Improved behavior on the Workbench. Tektronix emulation now allows saving IFF files, PostScript files, and printing bitmaps to the printer. Many other enhancements and bug fixes. Binary only. Author: Willy Langeveld

Fred Fish Disk 227

MidiLib A disk based library that permits sharing of the serial port by MIDI applications through a MIDI message routing and processing system. The midi utilities include a midi monitor to display incoming midi messages to the console, a routing utility, a midi library status utility, and more. V2.0, an update to FF101, and includes significant speed enhancements, new utilities to play with MIDI files, and updated utilities, documentation and examples. Binary only (source for examples and bindings however). Author: Bill Barton

PickPacket PickPacket gives you a visual display of the DosPacket structures that are sent to handlers, and lets you see the results. You can actually perform handler operations such as open files, read or write data, Examine or ExNext locks, and so forth, all by talking directly to the file system handler involved using PickPacket. V1.0, includes source. Author: John Toebes and Doug Walker

RexxArpLib A library which originally was supposed to be an ARexx interface to the ARP library. However, it has also become an interface to various Intuition functions, containing over 50 functions including a file requester, string/boolean requester, environment variable functions, simple message window, wildcard expander, etc. V2.3, an update to FF178. Binary only. Author: W.G.J. Langeveld

RexxMathLib A library which makes various high level math functions such as sin, tangent, log, etc. available in ARexx. V1.2 and 1.3, binary only. Author: W.G.J. Langeveld

Fred Fish Disk 228

Az A nice little text editor that is fast, simple to use, and very Amiga'ized. V1.40, binary only.Author: Jean-Michel Forgeas

Glib A text screen oriented librarian and editor for synths. Supports the TX81Z, DX100, DEP5, DW8000, and K-5. Includes source. Author: Tim Thompson, Steve Falco, and Alan Bland

JazzBench A drop-in multitasking replacement for WorkBench. It has more features that WorkBench and is fully multitasking (no more waiting for ZZZ clouds). It allows you to extend it, add your own menus, key shortcuts, etc. This is alpha version 0.8, binary only. Author: David Navas

Xoper Very comprehensive program to monitor and control system activity. Monitor cpu, memory usage, ports, interrupts, devices. Close windows, screens, show loaded fonts or last Guru code number. Clean up memory, flush unused libraries, devices, fonts. etc. and a whole bunch more! Spawns its own process. A very handy background task to have loaded. V1.3, an update to FF171. Assembly source included. Author: Werner Gunther

Fred Fish Disk 229

AlarmingClock A simple alarm clock program with a very alarming "ring", particularly if you hook it up to your stereo and turn up the volume. Includes source. Author: Brian Neal

DrawMap A program for drawing representations of the Earth's surface. Can generate flat maps, mercator maps, a globe view, or an orbital view. Includes source. Author: Bryan Brown

Emporos You are living on the island of Emporos, where several countries exist. Your goal is to make one of these countries your own. There is only one way to do this, and you have to find it out. Binary only. Author: Roland Richter

esuoM A little screen hack that causes the mouse pointer to move in the opposite direction of the mouse. Includes source. Author: Rob Eisenhuth

LeftyMouse Swaps the functions of the left and right mouse buttons so that Lefties can use the mouse with their left hands. Includes source. Author: Rob Eisenhuth

Shuffle A basic screen shuffler. Re-defines the key combination Left-Amiga-M to push the FRONT screen to the back, instead of pushing the Workbench screen to the back. Includes source. Author: Rob Eisenhuth

Sim A simulator for register-transfer nets, which are used to describe hardware systems. This version also provides a compiler to define new devices in addition to Sim's internal devices. Version 4.0, binary only. Author: Gotz Muller

Fred Fish Disk 230

AskTask Allows you to examine various bits of the task structures of all tasks in the system, from the lists attached to ExecBase. Diskplays priority, state, flags, stack, signals, etc. You can also remove tasks, change the priority of a task, or send arbitrary signals to a task. Version 2/4/89, includes source. Author: J. Bickers

Fedup A random access, byte oriented file-editor that gives you the option of viewing and editing any file (binary or ASCII) using either ASCII or hexadecimal notation. Version 2.1, binary only. Author: Martin Lindemann

FileIt A simple database program, written in DRACO. It is meant to be portable, thus it does not use any of Intuition's facilities. Version 1.0, includes source. Author: John Davis

NComm A communications program based on Comm version 1.34, by DJ James, with lots of very nice enhancements. Also includes several auxiliary programs such as AddCall, CallInfo, IbmIso, PbConvert, and ReadMail. This is version 1.8, binary only. Author: DJ James, Daniel Bloch, Torkel Lodberg, et al.

PrivHndlr A privilege violation handler for the 68010 cpu. Like Decigel, but survives a reboot so you can use it with copy protected programs that run from boot. Version 3, includes source in assembly code. Author: John Veldthuis

Quattro Another Tetris like program. Has three levels of play difficulty, sound effects, a 43-color background, next stone preview, and joystick or numberpad control. Version 1.0, binary only, source available from author. Author: Karl-Erik Jenss

Fred Fish Disk 231

Diff Yet another diff program. This one implements the algorithm from Communications of the ACM, April 78. It produces output which might be considered to be a little more user friendly than the standard Unix style diff programs. Includes source. Author: Donald C. Lindsay

File A program that recognizes various types of files and prints what that type is. Recognizes font files, icon files, executable files, standard object files, compressed files, command scripts, C source, directories, iff files, LaTeX source, modula II source, arc files, shell commands and scripts, TeX source, dvi files, uuencoded files, yacc files, zoo archives, etc. Version 1.0, includes source. Author: Edwin Hoogerbeets

NoClick2 A program which silences the clicking of empty drives on the B2000 under AmigaDOS 1.3. It should also work on an A500. Binary only, source available from author. Author: Norman Iscove

Plot A package for making 2D and 3D plots conveniently. AG Baxter wrote the intuition interface program (Plot) and Tim Mooney wrote the MultiPlot and ThreeDPlot programs, which are called from Plot. This is version 1.2 and includes source to Plot. Author: AG Baxter, Tim Mooney

Sed This is the GNU sed (stream editor) program, ported to the Amiga. Sed copies the named files, or the standard input, to its standard output, while performing certain editing operations specified in underlined, or inverse fonts. Version 1.8, includes source in Modula-II and assembly code. Author: Fridtjof Siebert

NetWork Another program in the long tradition of "screen hacks" for the Amiga. Won't spoil the surprise by saying what it does. Version 1.0, includes source in Modula-II. Author: Fridtjof Siebert

PrintIt A program to print IFF pictures on Epson compatible 9-pin printers. Prints in many resolutions, with many ways to convert color pics to black and white. Version 1.0, includes source in Modula-II. Author: Fridtjof Siebert

WBPic Replaces Workbench's color 0 with an IFF hires non-interlaced picture, in 2 or 4 colors. Version 1.0, includes source in Modula-II. Author: Fridtjof Siebert

XHair Replaces the mouse pointer with a screen wide crosshair, which is useful for positioning things vertically or horizontally. Version 1.0, includes source in Modula-II. Author: Fridtjof Siebert

Fred Fish Disk 235

CalcKey A basic four function, memory resident, pop-up calculator which uses only about 24K of memory and can automatically type the answer to any calculation into the program you were using when it was popped up. Version 1.0, binary only, shareware. Author: Craig Fisher

Ct An Amiga program to display images from a CT scanner, along with several new interesting sample images of scans of real people. The display software, though it has a primitive user interface, is quite powerful, including functions like convolutions, averaging, laplacians, unsharp masking, edge detection, gradients, etc. This is version 2.2, an update to the version on disk 137. Binary only. Additional image disks available from author. Author: Jonathan Harman

MirrorWars A new game featuring sound, title music, and two player mode. You fight your opponent via laser rays, but beware of the mirrors reflecting your shots. Binary only. Author: Oliver Wagner

Fred Fish Disk 236

AmigaBench Optimized Amiga assembly versions of the Dhrystone benchmark. Includes 68000 and 68020 versions. Author: Al Aburto

DiskHandler A sample implementation of a file system that reads and writes 1.2 format diskettes. Includes source. Author: Software Distillery

Heart3D A program to find left ventricle outlines in the output of an Imatron CT scanner, and display wireframe animations of the beating heart. Includes several sample CT scan outputs. Binary only. Author: Jonathan Harman

Ls Version 3.1 of the popular UNIX style directory lister. This is an update to version 2.0 from disk 178, and includes some bug fixes, support for multiple wildcard pathnames, quicker sorting, a best-fit output, new output width and height options, and some other new features. Includes source. Author: Justin V. McCormick.

Proc Example program of how to create a ful-fledged DOS process without needing to call LoadSeg first. Based on an idea presented at BADGE. Includes source. Author: Leo Schwab

XprZmodem An Amiga shared library which provides ZModem file transfer capability to any XPR-compatible communications program. Version 1.0, includes source. Author: Rick Huebner

Fred Fish Disk 237

CLIPrint An example of printing to the CLI from assembly code. Includes source (of course). by: Jeff Glatt

CType Another text file reader, but this one is small, reasonably fast, and includes bi-directional scrolling, search, go to a given percentage, and printing capabilities. Version 1.0, includes source in assembly. Author: Bill Nelson.

StripCR This little program just makes a text file ready for use with AmigaDOS, with only LineFeed characters (LF) to mark the end of a line. If you feed it a file with ONLY Carriage Return characters (CR), (from a Macintosh for example) it will replace them with the LF character and, if the file requires no changes, then it does not get changed, includes source in assembly. by: Bill Nelson

PlusCR Companion program to StripCR, it reverses the procedure. PlusCR produces a file ready for use on systems which require both the CR and LF characters to mark the end of a line (such as those running MS-DOS for example, includes source in assembly. Author: Bill Nelson

StripLF Completes the set of StripCR and PlusCR. It will change an LF only file into a CR only file. If used in combination with StripCR and PlusCR, it completes a text file conversion system, includes source in assembly. Author: Bill Nelson

CLS Clear Screen Command made for the purposes of being SMALL, and thus not wasteful of memory when made memory resident, it consists of 96 bytes of memory on disk! Includes source in assembly. Author: Bill Nelson

DPlot A simple display program for experimental data, with the goals of supporting paging through lots of data and providing comfortable scaling and presentation. Version 1.0, source available from author. Author: A. A. Walma

ILBMLib A shared library (ilbm.library) to read/write IFF files, derived from the EA IFF code, along with various enhancements. Includes examples of using the library from C code, assembly code, or BASIC, along with source for examples and interface code. Author: Jeff Glatt

ParOut Shows how to allocate and communicate directly with the parallel port hardware from an assembly language program. Includes source. Author: Jeff Glatt (original C code by Phillip Lindsay)

Speed A performance benchmark useful for comparing Amiga processing speeds. Performs 10000 iterations of some selected groups of 68000 instructions while using the DateStamp time function to record how many ticks it takes to complete. This timed duration is then compared against two known prestored times, one for a stock A2000 Amiga and one for an A2620 enhanced A2000. A relative comparison is calculated and displayed. Version 1.0, includes source in assembly language. Author: Jez San

Fred Fish Disk 238

CWDemo Demo version of a pop-up utility to control the color register assignments of Intuition custom screens. Version 3.1, binary only. Author: Kimbersoft

DMouse A versatile screen & mouse blanker, auto window activator, mouse accelerator, popcli, pop window to front, push window to back, etc, widget. Includes DLineArt, a screen blanker replacement program for use with DMouse. This is DMouse version 1.20, an update to version 1.10 on disk 168/169. Includes source. Author: Matt Dillon

LabelPrint A program that allows you to easily print labels for your disks. This is version 2.5, an update to version 1.9 from disk 210. Shareware, binary only (source available from author). Author: Andreas Krebs

NGC Yet another virus check program. Checks the bootblock on all inserted floppy disks and reports nonstandard ones. Checks the jump tables of all resident libraries and devices and reports suspicious entries. Version 1, includes source in assembly. Author: Ulf Nordquist

Pyth A program to draw the Tree of Pythagoras. Version 1.1, includes source. Author: Andreas Krebs

Steinschlag A tetris like game (Steinschlag means "Falling Rock") submitted by the author. This is version 1.8, an update to version 1.5 from disk 221. Binary only. Author: Peter Handel

Fred Fish Disk 239

FF 239 contains Forth programs from the JGoodies #1 disk, from Delta Research (the makers of JForth Professional 2.0). All of the material has been placed into a subdirectory (JGoodies). Below is a listing of subdirectories under JGoodies, and their contents.

Brunjes Various tools submitted together by the author. StringPkg is string package for both Forth style and NUL terminated strings. Date&Time are handy tools for getting and printing formatted date and time. Utils are utilities used by the other files. CursorControl is an example of moving the text cursor. SpaceOrEscape is a handy word for pausing or stopping program output. Includes source code. Author: Roy Brunjes

Evolution This program graphically simulates the evolution of a species of "bugs", the insect kind. Bugs, represented by moving blobs, eat bacteria represented by single pixels. They mutate, compete for food, reproduce and pass their mutations to their offspring. Fascinating example of graphics and software simulation. Standalone image and source code. Author: Russel Yost

FFT Highly optimized Fast Fourier Transform tools for digital signal processing. The FFT can be used to compute the frequency spectrum of a complex signal. It is useful in a variety of different applications. Floating point and integer versions. Mixture of high level and assembly language code. Includes source (requires JForth). Author: Jerry Kallaus

Guru Handy "guru" number interpreter (well, handy after reboot anyway!). Tells you what "81000009" means, for example. CLI usage only. Standalone image with readme file. Source code included. Author: Mike Haas

H2J Converts 'C' style '.h' include files to JForth style '.j' files. Useful when developing interfaces to new Amiga libraries like ARP, etc. Standalone image and source code. Author: Phil Burk

HAMmmm2 Graphics hack that displays moving lines in a HAM screen for a hypnotic effect. Uses sound tools from HMSL, if available, for a drone sound that corresponds to the graphics image. Standalone image and source code. Author: Phil Burk

HeadClean This program, combined with a fibre cleaning disk, can be used to clean the heads on your disk drives. Source code examples of accessing the Trackdisk device, and using gadgets are included. Standalone image with source code. Shareware. Version 2.0. Author: Phil Burk

JustBeeps Simple example of using Audio and Timer devices. Plays a series of beeps whose pitches are based on a just intoned tuning system. Standalone image with source code. Author: Phil Burk

Mandelbrot A fast Mandelbrot rendering program that uses some of the mathematical properties of the Mandelbrot set to greatly reduce the drawing time. Demonstrates graphics programming, assembly language, menus and IFF file I/O. Standalone image with source

the command line script or in a scriptfile. Version 1.02, includes source. Author: Unknown, ported to Amiga by Edwin Hoogerbeets

Fred Fish Disk 232

BallyIII Amiga port of the former arcade game named Click. This version fixes some minor bugs and is faster than the previous versions. This is version III, an update to the version released on disk 221. Binary only, shareware. Author: Oliver Wagner

Dbug Machine independent macro based C debugging package. Provides function trace, selective printing of internal state information, and more. This is an update to the version released on disk 102, and now includes a machine independent stack use accounting mechanism. Includes source. Author: Fred Fish; profiling support by Binayak Banerjee

ReSourceDemo A demo version of ReSource, an interactive disassembler for the Amiga. This is a complete version except that the "save" features have been disabled. This is version 3.06, an update to version 0.36 from disk 192. Binary only. Author: Glen McDiarmid

Fred Fish Disk 233

Brik A general purpose program that calculates both text and binary cyclic redundancy codes (CRCs). Text mode CRCs calculated by brik are portable across systems for files that are in the usual text format on each system. Binary mode CRCs are portable for files that are moved from system to system without any change. Brik can be used to verify and update an embedded checksum header in files. It runs under MS-DOS, UNIX system V, BSD UNIX, VAX/VMS, and AmigaDOS. This is version 2.0 and includes source. Author: Rahul Dhesi

CacheCard An accessory to SetCPU for use with A2620 cards or 68030 systems. It modifies the MMU table set up by SetCPU to selectively control caching for each expansion card. It's also an example of how an accessory program can track down and modify the SetCPU MMU table without having to read all kinds of MMU registers and figure it out for yourself. Version 1.00, includes source. Author: Dave Haynie

CrcLists Complete CRC check files for disks 001-231 using the brik program also on this disk. These were made directly from my master disks. I have switched to brik, from the crc program used to make the lists on disks 133, 146, and 173, because it has more features and because source is available. Author: Fred Fish

Fred Fish Disk 234

KwikBackUp A harddisk backup program that writes data track by track onto multiple floppy disks. Uses the archive bit, saves and restores comments and protection flags, and skips over bad spots during restore. Version 1.0, includes source in Modula-II. Author: Fridtjof Siebert

MuchMore Another program like "more", "less", "pg", etc. This one uses its own screen to show the text using a slow scroll. Includes built-in help, commands to search for text, and commands to print the text. Works with PAL or NTSC, in normal or overscan modes. Supports 4 color text in bold, italic, Textra This easy-to-use text editor allows multiple windows, and provides a simple mouse driven interface. Those familiar with the "Macintosh style" editors will be comfortable with Textra's Cut, Copy and Paste commands. Standalone image. Documentation included. No source code. Author: Mike Haa

Fred Fish Disk 240

CrossDOS A "tryware" version of a mountable MS-DOS file system for the Amiga. This is a software product that allows you to read and write MS-DOS/PC-DOS and Atari ST formatted disks (Version 2.0 or higher) directly from AmigaDOS. This tryware version is a "read only" version, which does not allow any writes to the disk. A fully functional version is available for a very reasonable price from CONSULTRON. Version 3.02, binary only. Author: CONSULTRON, Leonard Poma

Dis An AmigaDOS shareable library which implements a symbolic single-instruction disassembler for the MC68000 family and a program which uses the library to disassemble/dump AmigaDOS object files, making full use of symbolic and relocation information. Includes source code in Draco. Author: Chris Gray

DM-Maps IFF maps to the Dungeon Master game. All 14 levels are included. Author: Unknown

MemLib A link library of routines to aid in debugging memory problems. Works with Lattice C 5.0 and possibly with earlier versions. It's features include trashing all allocated memory, trashing all freed memory, keeping track of freed memory with notification if it is written to, notification of memory freed twice or not at all, notification of overrunning or underrunning allocated memory, generation of low memory conditions for testing purposes, and identification of violations of memory use by filename and line number of the allocating routine. Includes source. Author: John Toebes and Doug Walker

RunBack Allows you to start a new CLI program and run it in the background, then closes the new CLI. This is version 6, an update to the version on disk 152 (the version on disk 214 appears to be on a different evolutionary path). This version compiles under Lattice with many optimizations enabled, and can be made resident. Includes source. Author: Rob Peck, Daniel Barrett, Greg Searle, Doug Keller

XprLib External file transfer protocol library. Document and code example for implementing external file transfer protocols using Amiga shared libraries. This is an update to the version included with the vlt program on disk 226. Author: Willy Langeveld

Fred Fish Disk 241

ASDG-rrd Extremely useful shareware recoverable ram disk. This AmigaDOS device driver implements a completely DOS compatible disk device in memory that survives resets, guru's, and crashes. An absolute must for those with lots of ram. This is an update to the version released on disk 58. It now works with up to 8Mb of memory. It was rewritten in assembly and is now faster and much smaller. Binary only. Author: Perry Kivolowitz, ASDG Inc.

CBBS The WORLI BBS system for use in amateur radio. Originally written for IBM-PC compatibles, it was ported to the Amiga by Pete Hardie. This is version 6.1c with source code. Author: Hank Oredson, the CBBS group, Pete Hardie

Fix68010 A program which patches executables that fail to run on machines equipped with an M68010, so that they no longer use the prohibited priviledged instructions. Binary only. Author: Gregor Brandt

Man A program similar to the UNIX "man" program. Displays information about a topic from manual pages. Does not include any database of topics, you have to supply your own. Version 1.2, includes source. Author: Garry Glendown

NoClick A program which silences the clicking of empty drives on the B2000 under AmigaDOS 1.3. It should also work on an A500. This is version 3.4, an update to the version on disk 231. Includes assembly source code. Author: Norman Iscove

Tiles A basic tile game like Shanghai or GunShy. A board is covered with a set of 144 tiles, 36 different sets of 4 identical tiles, each with a picture on it. The object is to remove all the tiles, 2 at a time, by matching identical tiles. Version 2.1, includes source in Modula II. Author: Todd Lewis

Fred Fish Disk 242

BootBlocks Detailed documentation on what a bootblock is and how it works, along with some sample bootblocks and a program to install a custom bootblock. Includes source for the sample bootblocks and the install program. Author: Jonathan Potter

Check4Mem Allows you to check for a specified amount of memory, with certain attributes, from a batch file. If the requirements are not met, a WARN returncode is generated. Version 2, includes source. Author: Jonathan Potter

CustReq A glorified ASK command for your startup-sequence. It generates a requester with the specified title, text, positive and negative gadgets (either of which can be the default), and an optional timeout value. Version 2, includes source. Author: Jonathan Potter

FileReq This is Jonathan's second version of a file requester, and is much more powerful than the one included on disk 204. Shareware, includes source. Author: Jonathan Potter

FullView A text viewer that uses gadgets at the bottom of the screen (thus can display text 80 columns wide), opens up to the full height of the workbench screen, has fast scrolling, and can work with compressed files (file compression program included). Shareware, binary only, source availablefrom author. Author: Jonathan Potter

Image-Ed An icon editor that allows you to draw and edit images up to 150 by 90, in up to 16 colors. Allows freehand drawing, empty or filled rectangles, ellipses, and triangles, lines curves, and polygons, copy, flip about x or y axis, stretching and condensing, flood fill and complement, text with selection and loading of font style, undo, magnified and normal sized images, and two active drawing screens at once. This is version 2.2, an update to version 1.9 on disk 211. Binary only, source available from author. Author: Jonathan Potter

JAR A shareware game (Jump And Run) using 3-D graphics. Your task is to collect the blue pills lying on the floors and steps, not to fall down or off the steps, and to avoid several monsters wandering about. You can collect various sorts of weapons to use against the monsters. Version 1.0, binary only, source available from author. Author: Andreas Ehrentraut

JPClock A short clock program that is just packed with features. This is version 1.2, an update to version on disk 204. Includes source. by: Jonathan Potter

PPrefs Preferable Preferences is a program designed to replace the standard preferences, that is shorter, more efficient, and easier to use. Binary only. Author: Jonathan Potter

PaletteReq An easy way to set the palette of any screen from your program. Includes source. by: JonathanPotter

PopInfo A small utility which "pops open" to give you information about the status of your devices and memory. This is version 3.1, an update to version 3.0 on disk 223. Includes source. Author: Jonathan Potter

ZeroVirus A fully integrated virus checker and killer, with bootblock save and restore features. Finds both bootblock and file based viruses. This is version 1.3, binary only. Author: Jonathan Potter

Fred Fish Disk 243

FragIt A dynamic memory thrasher for the Amiga. FragIt randomly allocates and deallocates psuedo-random size values of memory, ranging from 16 bytes to 50000 bytes by default. The result is an allocation nightmare, thousands of memory fragments are being created and destroyed continuously. This puts stress on the memory allocation routines of an application undergoing testing by simulating a very busy, highly fragmented memory environment. This is version 2.0, featuring many bug fixes, a full intuition interface, configuration settings via the icon, and more. Includes source. Author: Justin V. McCormick

ImageLab A program which performs image processing on IFF pictures. Includes standard image processing functions such as convolution, averaging, smoothing, enhancement, histograms, FFT's, etc. Also includes file conversion functions, a clipboard, and other useful functions. Version 2.2, binary only. Author: Gary Milliorn

LPE LaTeX Picture Editor is a graphical editor for producing "pictures" for the LaTeX system, which may be imported by LaTeX. You can draw boxes, dashed boxes, lines, vectors, circles, boxes with centered text, and plain text. This is version 1.0, binary only. Author: Joerg Geissler

NoClick A program which silences the clicking of empty drives on the B2000 under AmigaDOS 1.3. It should also work on an A500. This is version 3.5, a last minute update to version 3.4 on disk 241. Includes assembly source code. Author: Norman Iscove

Password A program which enhances your computer's security by making it complicated enough that users without your password will get discouraged trying to boot and use your system. This should keep out most casual or nontechnical users. Version 1.21p, binary only. Author: George Kerber

Pcopy An intuition based disk copier similiar to the resident "DiskCopy". This is version 2.0, a highly upgraded rewrite of the version on disk 151. It features high speed diskcopy with write-verify and data recovery from damaged tracks. A lot of effort has gone into making this copier friendly in its usage, as well in its multitasking properties. Binary only. Author: Dirk Reisig

SimGen This program will add a 2 or 4 color picture to your WorkBench screen. If the picture is digitized, it will look much like a genlock, hence the name SimGen (Simulated Genlock). Binary only. Author: Gregg Tavares

SuperLines A new lines demo with a realtime control panel that you can use to change various aspects of the action. Has 10 built in color palettes, support for things like color "smudge", color cycling, color "bounce", multiple resolutions, and can display either lines or boxes. This is version 1.0, binary only. Author: Chris Bailey

WarpUtil Warp (version 1.11), UnWarp (version 1.0), and WarpSplit (version 1.1). Warp reads raw filesystems and archives them into a compressed version in a normal file. UnWarp turns them back into filesystems. WarpSplit splits them up into smaller pieces on a track by track basis. Binary only. Author: SDS Software

Fred Fish Disk 244

BBChampion This is BootBlockChampionIII, a very nicely done program that allows you to load, save, and analyze any bootblock. This is version 3.1, binary only. Author: Roger Fischlin

BootIntro This program creates a small intro on the bootblock of any disk, which will appear after you insert the disk for booting. The headline can be up to 44 characters. The scrolling text portion can be up to 300 characters. This is version 1.2, an update to version 1.0 on disk 188. Binary only. Author: Roger Fischlin

FMC An alternative to the NcFastMem program. Uses a cute little switch gadget to turn fast memory on or off. Version 1.2, includes source in assembly code. Author: Roger Fischlin

SizeChecker Size checker uses a list of possible sizes of a file to check for unexpected changes in the size of those files. For example, it can be used to spot a link virus or to point out changes in the configuration of your system. With the appropriate comments added to your size list, you can check to see what version of the files you are using (1.2, 1.3, 1.4, ARP, etc). Version 1.0, binary only. Author: Roger Fischlin

TextDisplay A text display program, like "more" or "less", but about half the size and handles all screen formats (pal/ntsc, interlace/non-interlace, etc). This is version 1.52, an update to version 1.1 on disk 188. Binary only. Author: Roger Fischlin

XColor A program designed to change the colors of any screen. You can also add and subtract bitplanes in the screen, or convert the screen to black and white (grayscale). Handles HAM and EHB screens. Version 1.2, includes source in assembly code. Author: Roger Fischlin

To Be Continued.....

In Conclusion

Subscribe and More!

Amazing is now available by Visa and MasterCard
dial 1-800-345-3360
or use the form below.

Amazing Deal!

Name_____

Address_____

City_____ State_____ Zip_____

Charge my ☐Visa ☐MC #_____ Expiration Date _____

Signature_____

Telephone 1-800-345-3360

All Charges are subject to a $20.00 minimum (charges under $20.00 will receive a $2.00 service charge.)
PROPER ADDRESS REQUIRED. In order to expedite and guarantee your order, all large Public Domain Software orders, as well as most Back issue orders, are shipped by United Parcel Service. UPS requires that all packages be addressed to a street address for correct delivery.

One Year Of AC! Please circle the appropriate item: **New Subscription** **Renewal**
Our regular renewal now includes 12 monthly issues of **Amazing Computing** **plus** the Spring edition of *AC's GUIDE TO AMIGA*

☐ $28.00 U.S ☐ $44.00 Foreign Surface ☐ $36.00 Canada and Mexico
(air mail rates available on request)

Our New SuperSub!
12 monthly issues of **Amazing Computing** **PLUS** *AC's GUIDE TO AMIGA*
3 Product Guides! Spring, Fall, and Winter! A savings of $32.25 off the newsstand price.
Please remember, the Product Guides alone retail for $6.95 each!

☐ $36.00 U.S ☐ $52.00 Foreign Surface ☐ $44.00 Canada and Mexico
(air mail rates available on request)

Please circle any additional choices below:

Single Back Issues:
$5 each US,
$6 each Canada & Mexico,
$7 each Foreign Surface.

1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9
2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12
3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12
4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 4.10

Public Domain Software:
$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers (three disk minimum on all foreign orders)
Please circle your Public Domain Software choices below:

Amazing on Disk: A#1...Source & Listings V3.8& V3.9
A#2...Source & Listings V4.4
A#3...Source & Listings V4.5 & V4.6
A#4...Source & Listings V4.7 & V4.8
A#5...Source & Listings V4.9
A#6...Source & Listings V4.10

InNOCKulation Disk: IN#1...Virus protection

AMICUS

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26

Fred Fish Disks

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 NA
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
77 78 79 NA 81 82 83 84 85 86 87 NA 89 90 91 92 93 94 95
96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152
153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190
191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209
210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228
229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244

(NA Denotes disks removed from the collection)

Please complete this form and mail with check, money order or credit card information to:

PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722-0869

Please allow 4 to 6 weeks for delivery

MasterCard®

VISA®

Subscription: $_____

Back Issues: $_____

PDS Disks: $_____

Total: $_____

Subscribe and More!

Amazing is now available by Visa and MasterCard
dial 1-800-345-3360
or use the form below.

Amazing Deal!

Name_____

Address_____

City_____ State_____ Zip_____

Charge my ☐Visa ☐MC #_____ Expiration Date _____

Signature_____ **Telephone 1-800-345-3360**

All Charges are subject to a $20.00 minimum (charges under $20.00 will receive a $2.00 service charge.)
PROPER ADDRESS REQUIRED. In order to expedite and guarantee your order, all large Public Domain Software orders, as well as most Back issue orders, are shipped by United Parcel Service. UPS requires that all packages be addressed to a street address for correct delivery.

One Year Of AC! Please circle the appropriate item: **New Subscription** **Renewal**
Our regular renewal now includes 12 monthly issues of **Amazing Computing**
plus the Spring edition of *AC's GUIDE TO AMIGA*
☐ $28.00 U.S ☐ $44.00 Foreign Surface ☐ $36.00 Canada and Mexico
(air mail rates available on request)

Our New SuperSub!
12 monthly issues of **Amazing Computing** **PLUS** *AC's GUIDE TO AMIGA*
3 Product Guides! Spring, Fall, and Winter! A savings of $32.25 off the newsstand price.
Please remember, the Product Guides alone retail for $6.95 each!
☐ $36.00 U.S ☐ $52.00 Foreign Surface ☐ $44.00 Canada and Mexico
(air mail rates available on request)

Please circle any additional choices below:

Single Back Issues:
$5 each US,
$6 each Canada & Mexico,
$7 each Foreign Surface.

1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9			
2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	2.10	2.11	2.12
3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9	3.10	3.11	3.12
4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	4.9	4.10		

Public Domain Software:
$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers (three disk minimum on all foreign orders)
Please circle your Public Domain Software choices below:

Amazing on Disk: A#1...Source & Listings V3.8& V3.9 A#2...Source & Listings V4.4
A#3...Source & Listings V4.5 & V4.6 A#4...Source & Listings V4.7 & V4.8
A#5...Source & Listings V4.9 A#6...Source & Listings V4.10

InNOCKulation Disk: IN#1...Virus protection

AMICUS

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26												

Fred Fish Disks

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	NA
58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
77	78	79	NA	81	82	83	84	85	86	87	NA	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114
115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133
134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152
153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171
172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228
229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244			

(NA Denotes disks removed from the collection)

Please complete this form and mail with check, money order or credit card information to:

PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722-0869

Please allow 4 to 6 weeks for delivery

Subscription: $_____

Back Issues: $_____

PDS Disks: $_____

Total: $_____

Amazing COMPUTING Reader Service Card

Want to know more about Amazing Computing Advertisers and their products? Here's your chance! Complete the card below and drop it in the mail (U. S. only). We will process your request and forward it to the appropriate advertiser. Quick, easy, and efficient—just like all the services of Amazing Computing: Your Original ***AMIGA*** Monthly Resource.

Amazing COMPUTING Reader Service Card

AC Oct. '89 Valid Until 12/31/89
see page 96 for reference numbers

Name _____
Street _____
City _____ ST. _____ ZiP _____
Country _____

A. Do you own an Amiga?
O 1. Amiga 1000 O 4. Soon
O 2. Amiga 500 O 5. Not Yet
O 3. Amiga 2000 O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
O 1. Amiga 500 O 6. Printer
O 2. Amiga 2000 O 7. Modem
O 3. Memory Expansion O 8. Music Tool
O 4. Hard Drive O 9. Video Product
O 5. IBM Emulators O 10. Other
(Sidecar or Bridgeboard)

C. What Amiga software product do you plan to buy next?
O 1. 'C' Language O 8. Spreadsheet
O 2. Forth Language O 9. Database
O 3. Modula-2 Language O 10. Financial
O 4. Assembly Language O 11. Video
O 5. BASIC Language O 12. Graphics
O 6. Entertainment O 13. Music
O 7. Telecommunications O 14. Other

D. Where do you buy your Amiga products?
O 1. Local Amiga Dealer O 3. Manufacturer
O 2. Discount Department Store O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

F. Which articles would you like to see more of in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

G. Are you a subscriber to Amazing Computing?
O 1. Yes O 2. No

Oct '89 1

101	102	103	104	105	221	222	223	224	225
106	107	108	109	110	226	227	228	229	230
111	112	113	114	115	231	232	233	234	235
116	117	118	119	120	236	237	238	239	240
121	122	123	124	125	241	242	243	244	245
126	127	128	129	130	246	247	248	249	250
131	132	133	134	135	251	252	253	254	255
136	137	138	139	140	256	257	258	259	260
141	142	143	144	145	261	262	263	264	265
146	147	148	149	150	266	267	268	269	270
151	152	153	154	155	271	272	273	274	275
156	157	158	159	160	276	277	278	279	280
161	162	163	164	165	281	282	283	284	285
166	167	168	169	170	286	287	288	289	290
171	172	173	174	175	291	292	293	294	295
176	177	178	179	180	296	297	298	299	300
181	182	183	184	185	301	302	303	304	305
186	187	188	189	190	306	307	308	309	310
191	192	193	194	195	311	312	313	314	315
196	197	198	199	200	316	317	318	319	320
201	202	203	204	205	321	322	323	324	325
206	207	208	209	210	326	327	328	329	330
211	212	213	214	215	331	332	333	334	335
216	217	218	219	220	336	337	338	339	340

Amazing COMPUTING Reader Service Card

AC Oct. '89 Valid Until 12/31/89
see page 96 for reference numbers

Name _____
Street _____
City _____ ST. _____ ZiP _____
Country _____

A. Do you own an Amiga?
O 1. Amiga 1000 O 4. Soon
O 2. Amiga 500 O 5. Not Yet
O 3. Amiga 2000 O 6. Just Looking

B. What Amiga hardware product do you plan to buy next?
O 1. Amiga 500 O 6. Printer
O 2. Amiga 2000 O 7. Modem
O 3. Memory Expansion O 8. Music Tool
O 4. Hard Drive O 9. Video Product
O 5. IBM Emulators O 10. Other
(Sidecar or Bridgeboard)

C. What Amiga software product do you plan to buy next?
O 1. 'C' Language O 8. Spreadsheet
O 2. Forth Language O 9. Database
O 3. Modula-2 Language O 10. Financial
O 4. Assembly Language O 11. Video
O 5. BASIC Language O 12. Graphics
O 6. Entertainment O 13. Music
O 7. Telecommunications O 14. Other

D. Where do you buy your Amiga products?
O 1. Local Amiga Dealer O 3. Manufacturer
O 2. Discount Department Store O 4. Mail Order

E. Which type of articles do you like to see in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

F. Which articles would you like to see more of in Amazing Computing?
O 1. 'C' Language O 10. Programming How To's
O 2. Forth Language O 11. Business How To's
O 3. Modula-2 Language O 12. Video Articles
O 4. Assembly Language O 13. Graphics Articles
O 5. BASIC Language O 14. Music Articles
O 6. Game Reviews O 15. Hardware How To's
O 7. Business Reviews O 16. PDS Updates
O 8. Hardware Product Reviews O 17. Interviews
O 9. Software Product Reviews O 18. Other

G. Are you a subscriber to Amazing Computing?
O 1. Yes O 2. No

Oct '89 2

101	102	103	104	105	221	222	223	224	225
106	107	108	109	110	226	227	228	229	230
111	112	113	114	115	231	232	233	234	235
116	117	118	119	120	236	237	238	239	240
121	122	123	124	125	241	242	243	244	245
126	127	128	129	130	246	247	248	249	250
131	132	133	134	135	251	252	253	254	255
136	137	138	139	140	256	257	258	259	260
141	142	143	144	145	261	262	263	264	265
146	147	148	149	150	266	267	268	269	270
151	152	153	154	155	271	272	273	274	275
156	157	158	159	160	276	277	278	279	280
161	162	163	164	165	281	282	283	284	285
166	167	168	169	170	286	287	288	289	290
171	172	173	174	175	291	292	293	294	295
176	177	178	179	180	296	297	298	299	300
181	182	183	184	185	301	302	303	304	305
186	187	188	189	190	306	307	308	309	310
191	192	193	194	195	311	312	313	314	315
196	197	198	199	200	316	317	318	319	320
201	202	203	204	205	321	322	323	324	325
206	207	208	209	210	326	327	328	329	330
211	212	213	214	215	331	332	333	334	335
216	217	218	219	220	336	337	338	339	340

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 36 FALL RIVER, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Amazing COMPUTING
P.O. Box 869
Fall River, MA 02722-0869

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 36 FALL RIVER, MA

POSTAGE WILL BE PAID BY ADDRESSEE

Amazing COMPUTING
P.O. Box 869
Fall River, MA 02722-0869

Please Mail to:

Amazing COMPUTING

P.O. Box 869
Fall River, MA 02722-0869

Place this order form in an envelope with your Check or Money Order.

Amazing Deal!

Name_____

Address_____

City_____ State_____ Zip_____

Charge my ☐Visa ☐MC #_____ Expiration Date _____

Signature_____

Telephone 1-800-345-3360

All Charges are subject to a $20.00 minimum (charges under $20.00 will receive a $2.00 service charge.)
PROPER ADDRESS REQUIRED. In order to expedite and guarantee your order, all large Public Domain Software orders, as well as most Back issue orders, are shipped by United Parcel Service. UPS requires that all packages be addressed to a street address for correct delivery.

One Year Of AC! Please circle the appropriate item: **New Subscription** **Renewal**

Our regular renewal now includes 12 monthly issues of **Amazing Computing** **plus** the Spring edition of *AC's GUIDE TO AMIGA*

☐ $28.00 U.S ☐ $44.00 Foreign Surface ☐ $36.00 Canada and Mexico

(air mail rates available on request)

Our New SuperSub!

12 monthly issues of **Amazing Computing** **PLUS** *AC's GUIDE TO AMIGA*

3 Product Guides! Spring, Fall, and Winter! A savings of $32.25 off the newsstand price. Please remember, the Product Guides alone retail for $6.95 each!

☐ $36.00 U.S ☐ $52.00 Foreign Surface ☐ $44.00 Canada and Mexico

(air mail rates available on request)

MasterCard ®

VISA ®

Subscription: $_____

Back Issues: $_____

PDS Disks: $_____

Total: $_____

Please circle any additional choices below:

Single Back Issues:
$5 each US,
$6 each Canada & Mexico,
$7 each Foreign Surface.

1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9
2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12
3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11 3.12
4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 4.10

Public Domain Software:

$6.00 each for subscribers (yes, even the new ones!)
$7.00 each for non subscribers **(three disk minimum on all foreign orders)**

Please circle your Public Domain Software choices below:

Amazing on Disk:
A#1. . .Source & Listings V3.8& V3.9
A#2. . .Source & Listings V4.4
A#3. . .Source & Listings V4.5 & V4.6
A#4. . .Source & Listings V4.7 & V4.8
A#5. . .Source & Listings V4.9
A#6. . .Source & Listings V4.10

InNOCKulation Disk: IN#1...Virus protection

AMICUS

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26

Fred Fish Disks

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 NA
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
77 78 79 NA 81 82 83 84 85 86 87 NA 89 90 91 92 93 94 95
96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152
153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190
191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209
210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228
229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244

(NA Denotes disks removed from the collection)

Please complete this form and mail with check, money order or credit card information to:

PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722-0869

Please allow 4 to 6 weeks for delivery

Variable Dither - Computed internally at 30 bits per pixel (over one billion colors). Gives you over 100,000 apparent colors on screen.

Unmatched.

Super BitMaps with Auto-Scrolling - Real-time scrolling on up to 1024 pixels high or wide image with full overscan display.

Sophisticated.

Flexible.

Flexible Text Rendering - Allows for anti-aliased fonts, Rainbow Fonts and Transparent Fonts and more.

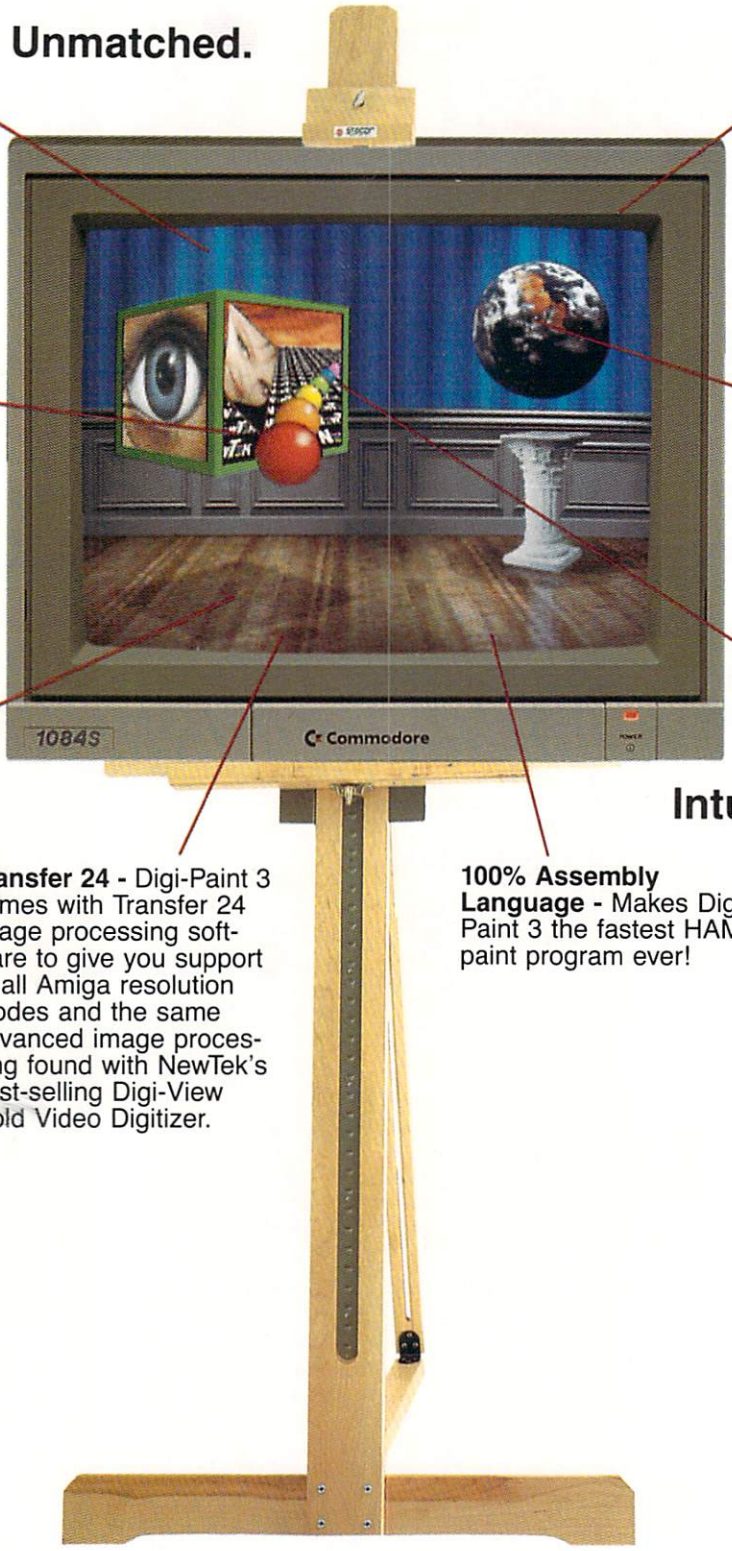
Colorize - Play Ted Turner and add color to black-and-white images or change colors on already colored images.

Revolutionary.

Powerful.

User-Controllable Transparency - Allows real time control of the amount of transparency and the location of the light source.

Texture Mapping with Anti-Aliasing - Gives you super-fast warping and stretching of any image.



Intuitive.

Unlimited.

Transfer 24 - Digi-Paint 3 comes with Transfer 24 image processing software to give you support of all Amiga resolution modes and the same advanced image processing found with NewTek's best-selling Digi-View Gold Video Digitizer.

100% Assembly Language - Makes Digi-Paint 3 the fastest HAM paint program ever!

The Ultimate Paint Program:

DIGI·PAINT™ 3

For more information call NewTek at 800-843-8934 or 913-354-1146

NEWTEK INCORPORATED

Circle 102 on Reader Service card.